Inhoud | Contents

Artikelen | Features

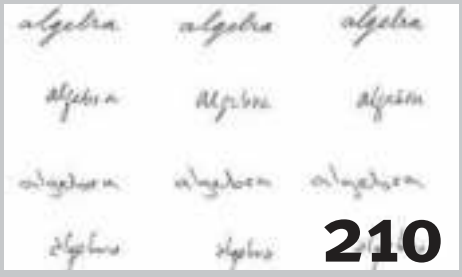
Machine Learning

Onder redactie van Marco Wiering, Jim Portegies en Sander Bohté

Rubrieken | Departments

Uitgelicht

Martijn van Otterlo belicht ethische vraagstukken omtrent kunstmatige intelligentie.



Lambert Schomaker, hoogleraar kunstmatige intelligentie, wordt geïnterviewd door Nicos Starreveld.

Colofon

Het Nieuw Archief voor Wiskunde is een uitgave van het Koninklijk Wiskundig Genootschap en verschijnt vier maal per jaar. Het tijdschrift richt zich op eenieder die zich beroepsmatig met wiskunde bezighoudt, als academisch of industrieel onderzoeker, student, leraar, journalist of beleidsmaker. Het stelt zich ten doel te berichten over ontwikkelingen in de wiskunde in het algemeen en in de Nederlandse wiskunde in het bijzonder.

ISSN 0028-9825

Adres
Nieuw Archief voor Wiskunde
Centrum Wiskunde & Informatica
Postbus 94079, 1090 GB Amsterdam
tel. 020-5924288
redactie@nieuwarchief.nl
www.nieuwarchief.nl

Hoofdredactie
Raf Bocklandt (r.r.j.bocklandt@uva.nl)
Robbert Fokkink (r.j.fokkink@tudelft.nl)

Eindredactie/Bladmanager
Fred Drissen (fred@nieuwarchief.nl)

Redactie
Viktor Blåsjö (UU), Hans Cuypers (TU/e), Geertje Hek (UvA), Teun Koetsier (VU), Margriet Oomen (UL), Nicolaos Starreveld (UvA), Hans Sterk (TU/e), Mark Timmer (UT)

Problemenrubriek
Rob Eggermont (problems@nieuwarchief.nl)

Bureauredactie
Claartje Ottens

Illustraties
Ryu Tajiri

Vormgeving
Susanne Laws (omslag, begeleiding binnenwerk)
Kitty Molenaar (ontwerp)

Druk
Veldhuis Media

Advertenties
advertenties@nieuwarchief.nl

Koninklijk Wiskundig Genootschap
www.wiskgenoot.nl

Platform Wiskunde Nederland
www.platformwiskunde.nl

European Mathematical Society
www.euro-math-soc.eu

International Mathematical Union
www.mathunion.org

Koninklijk Wiskundig Genootschap

Het Koninklijk Wiskundig Genootschap (KWG) is een landelijke vereniging van beoefenaren van de wiskunde. In 1778 opgericht onder de zinspreuk: 'Een onvermoeide arbeid komt alles te boven' is het 's werelds oudste nationale wiskundegenootschap. Leden ontvangen het Nieuw Archief voor Wiskunde als onderdeel van hun lidmaatschap.

Bestuur
Jan Wiegerinck (UvA, voorzitter), Marije Elkenbracht (ABN AMRO, penningmeester), Sonja Cox (UvA, secretaris), Erik van den Ban (UU), Danny Beckers (VU), Theo van den Bogaart (HU), Marie-Colette van Lieshout (CWI), André Ran (VU), Mark Veraar (TUD)

Secretariaat
Koninklijk Wiskundig Genootschap, CWI
Postbus 94079, 1090 GB Amsterdam
wiskgenoot@wiskgenoot.nl
www.wiskgenoot.nl

Ledenadministratie KWG / Abonnementen
Postbus 1064, 7940 KB Meppel
admin@wiskgenoot.nl, tel. 085-0160252

Contributie
Tarieven per jaar (bij betaling per automatische incasso krijgt u € 2,50 korting):
– gewoon lid: € 92,50
– reciprociteitslid: € 72,50
– gepensioneerden en werklozen: € 47,50
– aio's, oio's, studenten: € 37,50 (max. 4 jaar)
– lid zonder Nieuw Archief: € 52,50
Studenten en pas afgestudeerden kunnen een jaar lang gratis lid worden.
Voor verzending van het Nieuw Archief naar het buitenland wordt een portotoeslag van € 10 in rekening gebracht.
Instituten in Nederland en buitenland kunnen zich abonneren op het Nieuw Archief voor Wiskunde voor € 100 (Nederland), € 110 (Europa) of € 112,50 (buiten Europa).

Members of foreign societies
Members of *SIAM*, the *Société Mathématique de France*, the *Gesellschaft für Angewandte Mathematik und Mechanik*, the *Deutsche Mathematiker-Vereinigung*, and of the *American*, *Australian*, *Belgian*, *Indian*, *London*, *Spanish*, and *South-African Mathematical Societies* living outside of the Netherlands pay € 72,50 instead of the full membership fee of € 92,50 a year. Conversely, these foreign societies, as well as the VvS+OR and the NVORWO, have reduced membership fees for KWG members. A postage charge of € 10 is added for members living abroad but in Europe, and a charge of € 12,50 for members living outside of Europe.

Exchange subscriptions
Koninklijk Wiskundig Genootschap Library
Centrum Wiskunde & Informatica
P.O. Box 94079, NL-1090 GB Amsterdam
KWG-exchange@cwi.nl

Informatie voor auteurs

Kopij
Het Nieuw Archief voor Wiskunde is een gerediger tijdschrift. Kopij dient per e-mail te worden aangeboden aan de hoofdredactie. Uitgebreide informatie en auteursinstructies kunt u vinden op www.nieuwarchief.nl.

Volgende nummers

nummer	*maand verschijnen*	*deadline kopij*
19(4)	december 2018	verstreken
20(1)	maart 2019	01-11-2018
20(2)	juni 2019	01-02-2019
20(3)	september 2019	01-05-2019

Instituutsleden

De publicaties van het Koninklijk Wiskundig Genootschap worden mede mogelijk gemaakt door de bijdragen van de volgende instituten.

Centrum Wiskunde & Informatica

Rijksuniversiteit Groningen

Universiteit van Amsterdam

Universiteit Leiden

Vrije Universiteit

Universiteit Utrecht

Technische Universiteit Eindhoven

Eurandom

Technische Universiteit Delft

Universiteit Twente

Radboud Universiteit Nijmegen

Freudenthal Instituut

Op het omslag

Dit themanummer gaat over 'Machine Learning', ofwel Machinaal Leren, het deelgebied van Kunstmatige Intelligentie dat zich bezighoudt met technieken waarmee computers kunnen leren van data. Foto: Alamy Stock Foto, RoboCup 2009, Graz.

Machine Learning

Vrij regelmatig wordt moderne *Artificial Intelligence*, of AI, in de media neergezet als een andere naam voor *Deep Learning*, ofwel de moderne noemer voor klassieke neurale netwerken opgeschaald naar de computerkracht van de 21ste eeuw. Wetenschappelijk is Deep Learning echter een deelgebied van *Machine Learning*, of Machinaal Leren, het deelgebied van Kunstmatige Intelligentie dat zich bezighoudt met technieken waarmee computers kunnen leren van data. Mediaverwarring lijkt de prijs van succes te zijn.

Dit neemt niet weg dat Machine Learning in bredere zin de afgelopen jaren een grote vlucht heeft genomen. In dit themanummer zijn zeer diverse bijdragen verzameld van Machine Learning-onderzoekers aan Nederlandse universiteiten en onderzoeksinstellingen.

Verschillende aspecten van Deep Learning passeren de revue en geven een snelle introductie: reinforcement learning dat agents laat leren door interactie met hun omgeving, vector space models voor de verwerking van natuurlijke taal, en biologisch geïnspireerde neurale netwerken. Ook statistische en logische aspecten van Machine Learning komen aan bod, in het kader van adaptive sensing, relaties tot speltheorie, en universele leeralgoritmes die nabootsen hoe mensen leren. Tot slot is er aandacht voor de belangrijke ethische aspecten van Kunstmatige Intelligentie. Bij elkaar een fraaie greep uit een bloeiend vakgebied.

Wiskunde speelt een belangrijke rol in de theorie en begrip van Machine Learning, een rol die waarschijnlijk alleen maar groter gaat worden omdat theorie voor *Deep Learning* nog grotendeels ontbreekt.

Is er nog meer? Uiteraard! Met de grote inbreng van bedrijven loopt Machine Learning ook voorop bij innovaties op het gebied van publiceren. Niet alleen verschijnen vrijwel alle bijdragen direct op arXiv, grote spelers als Google Deepmind, Facebook, Microsoft en Apple schrijven ook blogs over hun artikelen, gericht op een breder publiek voor een snel intuïtief begrip. De geïnteresseerde lezer kan dan ook direct doorsteken.

Marco Wiering, *Kunstmatige Intelligentie, Rijksuniversiteit Groningen*
Jim Portegies, *Wiskunde en Informatica, Technische Universiteit Eindhoven*
Sander Bohté, *Machine Learning, Centrum Wiskunde & Informatica, Amsterdam*
gastredacteuren

Agenda

| Upcoming Events

september 2018

10–13 september 2018

❑ **Statistical Inference for Stochastic Process Models in Weather and Climate Science**

Een workshop mede georganiseerd door Shota Gugushvili (Leiden) en Peter Spreij (Amsterdam).

plaats *Lorentz Center@Snellius, Leiden*

info *lorentzcenter.nl*

10–14 september 2018

❑ **NWO-JSPS Joint Seminar: Computations on Networks with a Tree-Structure**

Workshop mede georganiseerd door Hans Bodlaender, Erik Jan van Leeuwen, Tom van der Zanten (UU), Bart Jansen en Jesper Nederhof (TU/e).

plaats *Eurandom, Eindhoven*

info *eurandom.nl*

14 september 2018

❑ **Finale Nederlandse Wiskunde Olympiade**

De finale van de jaarlijkse landelijke wiskundewedstrijd voor leerlingen van havo en vwo.

plaats *Eindhoven*

info *wiskundeolympiade.nl*

18–21 september 2018

❑ **LeGO 2018: 14th International Workshop on Global Optimization**

Tweejaarlijkse workshop georganiseerd door Michael Emmerich, Andre Deutz, Sander Hille en Yaroslav Sergeyev.

plaats *Universiteit Leiden*

info *universiteitleiden.nl*

29 september 2018

❑ **Wiskundetoernooi**

Wiskundetoernooi voor teams van middelbare scholieren. Een team bestaat uit 4 of 5 leerlingen uit 5–6 vwo en een begeleider.

plaats *Radboud Universiteit, Nijmegen*

info *ru.nl/wiskundetoernooi*

29 september 2018

❑ **Junior Wiskunde Olympiade**

Jaarlijkse landelijke wiskundewedstrijd voor leerlingen van havo en vwo, georganiseerd door de VU in samenwerking met de Nederlandse Wiskunde Olympiade en W4Kangoeroe.

plaats *Vrije Universiteit, Amsterdam*

info *beta.vu.nl/nl/scholieren*

oktober 2018

1–3 oktober 2018

❑ **Multidimensional Queues, Risk, and Finance**

Workshop met als onderwerp stochastische modellen in de wachtrij- en de verzekeringstheorie, georganiseerd door Onno Boxma, Stella Kapodistria (TU/e) en David Koops (UvA).

plaats *Eurandom, Eindhoven*

info *eurandom.nl*

3–5 oktober 2018

❑ **43rd Woudschoten Conference 2018**

Een jaarlijkse conferentie van de Nederlands-Vlaamse Numerieke Analyse-groepen, georganiseerd door de Werkgemeenschap Scientific Computing (WSC).

plaats *Woudschoten, Zeist*

info *wsc.project.cwi.nl/events*

5 oktober 2018

❑ **Dag van de Leraar**

Ieder jaar is het op 5 oktober de internationale Dag van de Leraar. In het kader van deze dag organiseert de Onderwijscoöperatie de verkiezing Leraar van het Jaar.

plaats *overal*

info *dagvandeleraar.nl*

6 oktober 2018

❑ **CWI Open Dag**

CWI doet mee aan de Amsterdam Science Park Open Dag tijdens het Weekend van de Wetenschap.

plaats *CWI, Amsterdam*

info *cwi.nl*

8–12 oktober 2018

❑ **Molecular Simulations Meets Machine Learning and Artificial Intelligence**

Workshop mede georganiseerd door Shirin Faraji, Siewert-Jan Marrink en Niels Taatgen (RUG).

plaats *Lorentz Centrum@Snellius, Leiden*

info *lorentzcenter.nl*

13 oktober 2018

❑ **Symposium werkgroep geschiedenis**

24ste symposium van de werkgroep geschiedenis van de NVvW met als thema ‘Reken je rijk’. Over hoe en waarom er vroeger gerekend werd.

plaats *Vergadercentrum Domstad, Utrecht*

info *nvvw.nl*

Suggesties voor deze agenda zijn welkom.

Redactie:
Margriet Oomen en Nicolaos Starreveld
agenda@nieuwarchief.nl

16 oktober 2018
❑ **KNAW Symposium on Networks**
Symposium georganiseerd door Elle Uijtdewilligen (KNAW), Michel Mandjes (UvA) en Frank den Hollander (UL).
plaats *Openbare Bibliotheek, Amsterdam*
info *knaw.nl*

17–19 oktober 2018
❑ **MATTS 2018**
Workshop 'Mathematics Applied in Transport and Traffic Systems' is een vervolg op eerdere workshops over de wiskundige fundamenten van verkeersmodellering.
plaats *Science Centre, TU Delft*
info *tudelft.nl*

november 2018

3 november 2018
❑ **Jaarvergadering NVvW**
Jaarlijkse vergadering en studiedag van de Nederlandse Vereniging van Wiskundeleraren.
plaats *Ichthus College, Veenendaal*
info *nvvw.nl*

7 november 2018
❑ **Twentse Wiskunde Estafette**
Leerlingen uit 4 en 5 vwo doen in teams mee aan een uitdagende wiskundige wedstrijd.
plaats *Universiteit Twente, Enschede*
info *utwente.nl*

9 november 2018
❑ **Prijsuitreiking van de Nederlandse Wiskunde Olympiade**
De prijsuitreiking van de finale van de Nederlandse Wiskunde Olympiade.
plaats *Technische Universiteit Eindhoven*
info *wiskundeolympiade.nl*

12–14 november 2018
❑ **47ste Bijeenkomst Stochastici**
Workshop georganiseerd door Marie-Colette van Lieshout (CWI).
plaats *Congrescentrum De Werelt, Lunteren*
info *eurandom.nl*

16 november 2018
❑ **Wiskunde B-dag**
De Wiskunde B-dag is een 1-daagse opdracht voor op school voor teams uit 5 havo en 5/6 vwo met wiskunde B in hun profiel.
plaats *deelnemende scholen*
info *uu.nl/onderwijs/wiskunde-b-dag*

16 november 2018
❑ **Voorronde A-lympiade**
Voorronde van de jaarlijkse competitie voor leerlingen uit de bovenbouw havo/vwo met wiskunde A in hun pakket.
plaats *eigen school*
info *www.fi.uu.nl/alympiade*

29–30 november 2018
❑ **Diamant Symposium**
Tweedaags symposium georganiseerd door het wiskundecluster DIAMANT.
plaats *Hotel Van der Valk, Veenendaal*
info *leidenuniv.nl*

30 november 2018
❑ **TLL Herfstfestival 2018**
Festival van het Teaching & Learning Lab. Ervaren, uitproberen en zelf onderwijs maken binnen voortgezet en hoger onderwijs.
plaats *Buys Ballotgebouw, Universiteit Utrecht*
info *teachinglearninglab.nl*

december 2018

3–7 december 2018
❑ **Mathematical Modeling with Measures: Where Applications, Probability and Determinism Meet**
Een workshop mede georganiseerd door Sander Hille (UL).
plaats *Lorentz Center@Snellius, Leiden*
info *lorentzcenter.nl*

10–14 december 2018
❑ **Machine Learning and Reverse Engineering for Soft Materials**
Een workshop mede georganiseerd door Marjolein Dijkstra (UU).
plaats *Lorentz Centrum@Oort, Leiden*
info *lorentzcenter.nl*

januari 2019

14–16 januari 2019
❑ **LNMB-conferentie**
44ste conferentie over de wiskunde van besliskunde georganiseerd door het Landelijk Netwerk Mathematische Besliskunde (LNMB).
plaats *Congrescentrum De Werelt, Lunteren*
info *lnmb.nl*

21–23 januari 2019
❑ **18th Winter School Mathematical Finance**
Winterschool over financiële wiskunde met onder andere minicursussen van Jim Gatheral (The City University of New York) en Paolo Guasoni (Dublin City University).
plaats *Congrescentrum De Werelt, Lunteren*
info *staff.science.uva.nl/p.j.c.spreij*

21–31 januari 2019
❑ **Eerste ronde Wiskunde Olympiade**
De eerste ronde van de jaarlijkse landelijke wiskundewedstrijd voor leerlingen uit de onderbouw van havo en vwo.
plaats *op eigen scholen*
info *wiskundeolympiade.nl*

22 januari 2019
❑ **The Road to Reality**
Een workshop met als spreker Sir Prof. Roger Penrose georganiseerd door het Korteweg-de Vries Instituut voor Wiskunde en het Institute for Advanced Study (IAS Amsterdam).
plaats *IAS, Amsterdam*
info *ias.uva.nl*

februari 2019

1–2 februari 2019
❑ **Nationale Wiskunde Dagen**
De 25ste editie van de tweedaagse conferentie voor wiskundeleraren.
plaats *Leeuwenhorst, Noordwijkerhout*
info *uu.nl*

Nieuws

| News

Deze rubriek is een kroniek van wiskundige activiteiten in Nederland. Toekomstige activiteiten worden aangekondigd en van voorbije activiteiten wordt verslag gedaan. Wilt u uw aankondiging of verslag in deze rubriek geplaatst zien? Stuur ons dan uw bijdrage, zo mogelijk met illustratie. De redactie behoudt zich het recht voor berichten te weigeren of in te korten.
Redactie: Margriet Oomen en Nicolaos Starreveld
nieuws@nieuwarchief.nl

Jan Aarts overleden

Op donderdag 14 juni 2018 is prof. dr. Jan Aarts overleden. Johannes Michael Aarts werd in 1938 te Sittard geboren. Hij studeerde wiskunde aan de UvA en promoveerde aldaar in 1966 bij professor De Groot in de topologie. In 1967 werd hij aan de toenmalige TH Delft aangesteld als lector. In 1973 volgde zijn benoeming tot hoogleraar in de zuivere en toegepaste wiskunde.

Het onderzoek van Jan Aarts concentreerde zich op dimensietheorie en topologische dynamica. Daarnaast stond hij bekend als betrokken en bevlogen docent. Hij was ook gedurende lange tijd zeer actief voor de Wiskunde en Informatica Studievereniging 'Christiaan Huygens', waar hij erevoorzitter was.

Veel waardering is er ook voor Jan als het gaat om zijn inspanningen op bestuurlijk vlak. Zo was hij binnen de TU Delft voorzitter van de vakgroep Algemene Wiskunde, opleidingsdirecteur, decaan van de faculteit Wiskunde en Informatica en conrector. Op nationaal niveau was Jan lid van het bestuur van de wiskundeonderzoeksschool het Thomas Stieltjes Instituut en directeur van MasterMath, waarin de Nederlandse universiteiten samenwerken op het gebied van masteronderwijs. Ook heeft hij jarenlang meegewerkt aan de organisatie van de Nationale Wiskunde Dagen. Op 9 mei was er nog een lezingenmiddag ter gelegenheid van Jan zijn tachtigste verjaardag. *tudelft.nl*

Kwantumcomputers versus traditionele computers

Zijn er problemen die alleen een kwantumcomputer en niet een traditionele computer kan oplossen? Al in 1993 stelden informatici deze vraag, maar het antwoord kwam pas onlangs. De informatici Ran Raz van Princeton University en Avishay Tal van Standford University publiceerden 31 mei een artikel waarin bewezen wordt dat zo'n probleem inderdaad bestaat. Het vraagstuk is als volgt. Stel je hebt twee random number generators en elke generator produceert een rij getallen. Kan een computer bepalen of deze twee rijen getallen volledig onafhankelijk zijn of toch op een verborgen ma-

nier afhankelijk van elkaar zijn? Een kwantumcomputer kan dit probleem oplossen met slechts één hint. Voor een 'gewone' computer, zelfs voor een futuristische superieure, maar traditionele computer is dit onmogelijk, ook als de computer oneindig veel hints krijgt. Hiermee bewezen Raz en Tal dat BQP, de klasse van vraagstukken die een kwantumcomputer kan oplossen fundamenteel verschilt van PH, de klasse van vraagstukken die een traditionele computer kan verifiëren en/of oplossen. Hoewel kwantumcomputers in de praktijk nog niet gebruikt kunnen worden, is het werk van Raz en Tal een indrukwekkende stap voorwaarts op gebied van computationele complexiteitstheorie, een belangrijk vakgebied binnen de theoretische informatica. *quantamagazine.org*

Zeventienjarige Matthijs wint zilver op IMO

Bij de Internationale Wiskunde Olympiade in Cluj-Napoca in Roemenië, van 7 tot 14 juli, heeft de zeventienjarige Matthijs van der Poel uit IJsselstein een zilveren medaille behaald. Hij loste vier van de zes opgaven perfect op. Ook de vijf andere leerlingen uit het Nederlandse team vielen in de prijzen. Nils van de Berg, Jippe Hoogeveen, Jovan Gerbscheid en Thomas Chen wonnen allemaal een bronzen medaille en Szabi Buzogany verdiende een eervolle vermelding voor het foutloos oplossen van één van de zes opgaven. Matthijs van der Poel: "Het waren erg leuke opgaven om aan te werken en ook de rest van het programma was superleuk. Het is heel bijzonder om aan zo'n internationale wedstrijd deel te nemen en ik ben blij met het resultaat."

Het Nederlandse team werd begeleid door Birgit van Dalen (UL, ISW Hoogeland Naaldwijk), Quintijn Puite (TU/e, HU) en Jetze Zoethout (UU). *wiskundeolympiade.nl*

CWI-onderzoekers maken energienetwerk stabieler met wiskunde

Onderzoekers van Centrum Wiskunde & Informatica (CWI) hebben ontdekt hoe schommelingen van wind- en zonne-energie storingen in elektriciteitsnetwerken kunnen veroorzaken. Het wiskundig raamwerk ontwikkeld door de onderzoekers biedt ondersteuning bij het voorspellen van mogelijke stroomstoringen in het hoogspanningsnetwerk. Het wetenschappelijk artikel is gepubliceerd in *Physical Review Letters* op 21 juni 2018. Extreme weersomstandigheden kunnen ertoe leiden dat lijnen in energienetwerken overbelast raken en daardoor een storing krijgen. Een storing veroorzaakt een herverdeling van energiestromen, waarmee de druk op de overgebleven lijnen in het netwerk toeneemt. Dit kan meer storingen en zelfs zogenaamde black-outs veroorzaken. Het begrijpen van dit proces is van groot belang, omdat de transformatie naar een duurzame samenleving niet moet leiden tot een achteruitgang van de kwaliteit van het energietransport.

CWI onderzoekers Tommaso Nesti, Alessandro Zocca en Bert Zwart hebben onderzocht hoe storingen zich kunnen ontwikkelen onder de invloed van wind- en zonne-energie. Ze gebruikten daarvoor een analogie uit de statistische fysica, waardoor ze een groot energienetwerk met veel input van duurzame energie kunnen interpreteren als een interactief deeltjessysteem. Hierdoor kunnen de lijnen in het netwerk die het meest kwetsbaar zijn voor schommelingen in weerpatronen geïdentificeerd worden, evenals het meest waarschijnlijke scenario waarop deze storingen zich zullen verspreiden over het netwerk. Het blijkt dat verstoringen op de elektriciteitslijnen kunnen worden veroorzaakt door een cumulatief effect van kleine fluctuaties die zich voordoen in een groot geografisch gebied. Het aantal opeenvolgende storingen kan veel hoger uitvallen dan in voorspellingen gedaan door simpelere modellen, waarbij geen rekening gehouden werd met weerpatronen. *cwi.nl*

Wiskundigen ontkrachten vermoeden van Roger Penrose

Sir prof. Roger Penrose is een natuur- en wiskundige, hij is emeritus hoogleraar wiskunde aan de Universiteit van Oxford. Penrose heeft onder andere een significante bijdrage geleverd in de kosmologie. Hij heeft met Steven Hawking gewerkt, en een theorie ontwikkeld waarin hij de principes van algemene relativiteit toepast in zwarte gaten. Hij heeft voor het eerst stellingen over singulariteiten ge-

Corrigendum

On page 122 of the printed version of the June issue (NAW 5/19 nr. 2, 2018), by mistake, a portrait of Niels Abel has been placed instead of Évariste Galois.

formuleerd en bewezen. In de jaren zestig zagen wiskundigen dat de toepassing van algemene relativiteit in zwarte gaten niet altijd soepel liep, het bleek dat Einsteins vergelijkingen meerdere oplossingen hadden. Het bestaan van meerdere oplossingen betekent dat de evolutie van ruimte-tijd niet deterministisch is, en dat was erg verontrustend. Penrose probeerde deze inconsistentie in orde te maken en heeft een vermoeden gesteld, dat bekend is als de 'Cosmic censorship conjecture'. Twee wiskundigen, Mihalis Dafermos (Princeton University) en Jonathan Luk (Stanford University), hebben, ongeveer een jaar geleden, een artikel op arXiv geüpload waarin ze bewijzen dat Penroses vermoeden niet helemaal klopt en dat determinisme niet in gevaar is! Hun werk begint langzamerhand veel aandacht te krijgen. Een interessant artikel is op quantamagazine te vinden, het onderzoeksartikel is op arXiv te vinden en is nog niet gepubliceerd. *quantamagazine.org, arXiv.org*

Incomplete open cubes

Op de vijfde verdieping van het Museum of Modern Art in San Francisco (SF-MOMA) staat een kunstwerk van de Amerikaanse beeldhouwer, schilder en tekenaar Sol Lewitt. Een collectie van verschillende deelskeletten van de kubus genaamd 'incomplete open cubes'. Het is volgens de maker een zoektocht door alle 122 verschillende vormen van een onvolledige kubus, beginnend met drie ribben en eindigend met elf. Onlangs ontstond enige consternatie toen een bezoeker opmerkte dat twee van de kubussen identiek zijn. Het gaat om de skeletten met tien ribben op bijgaande foto. Heeft de kunstenaar een fout gemaakt? Is het een vervalsing? Is het een artistieke vrijheid die demonstreert dat minimal art niet altijd minimaal hoeft te zijn? SF-MOMA heeft de zaak in onderzoek. *Oakland Observer*

Topprestaties van Nederlandse studenten bij IMC

De International Mathematics Competition for University Students is een jaarlijkse wiskundewedstrijd voor wiskundestudenten die niet ouder dan 23 zijn. De IMC vindt ieder jaar plaats in Blagoevgrad, Bulgarije. Hoewel het een individuele wedstrijd is, doen de meeste studenten mee in een team van hun eigen universiteit. Dit jaar waren Nederlandse studenten zeer succesvol, ze hebben in totaal vier gouden, negen zilveren en vier bronzen medailles thuisgebracht! Pim Spelier van de UL was de beste Nederlandse deelnemer op de twintigste plaats. In de teamklassering zijn de UvA, de UL en de TU Delft op respectievelijk plaats 17, 30 en 43 geëindigd. *imc-math.org.uk*

Fieldsmedaille voor vier wiskundigen

Op 1 augustus zijn tijdens het ICM 2018 in Rio de Janeiro de Fieldsmedailles uitgereikt. De Fieldsmedailles worden eens in de vier jaar toegekend aan twee tot vier wiskundigen die jonger dan veertig jaar zijn. Dit jaar zijn er vier Fieldsmedailles uitgereikt, aan Caucher Birkar (Cambridge), Alessio Figalli (ETH), Peter Scholze (Bonn) en Akshay Venkatesh (Stanford). Birkar doet onderzoek in de algebraïsche meetkunde, Figalli is een expert in transporttheorie, die hij toepast in partiële differentiaalvergelijkingen, meetkunde en kansrekening. Scholze heeft een significante bijdrage geleverd in de algebraïsche meetkunde en de representatietheorie. Venkatesh onderzoekt de connecties tussen analytische getaltheorie, dynamica, topologie en representatietheorie. *mathunion.org*

Foto: AFP

Caucher Birkar, Alessio Figalli, Peter Scholze en Akshay Venkatesh

Marco Wiering
Department of Artificial Intelligence
University of Groningen
m.a.wiering@rug.nl

Reinforcement learning: from methods to applications

Reinforcement learning (RL) algorithms enable computer programs to learn from interacting with an environment. The goal is to learn the optimal policy that maximizes the long-term intake of a reward signal, where rewards are given to the agent for reaching particular environmental situations. The field of RL has developed a lot since the past decade. In this paper, Marco Wiering will first examine different decision making problems and RL algorithms. Then the combination of RL with different function approximators that can be used to solve large-scale problems will be described. After explaining several other ingredients of an RL system, a wide variety of different applications that can be fruitfully solved by RL systems will be covered.

Machine learning is a very hot research area within the larger field Artificial Intelligence (AI). The main idea of machine learning algorithms is to enable a computer program to optimize a particular performance metric using data or experiences to learn from. The field can be divided in three main areas: supervised learning, unsupervised learning and reinforcement learning. In supervised learning, a dataset with inputs and target outputs is given to a machine learning algorithm. The algorithm (with all of its tunable hyperparameters) then generates a model (or function) that can map inputs to their corresponding target outputs. An important issue in supervised learning is to generate a trained model that also performs well for new, unseen, inputs, which means the model should generalize over the whole input space.

In unsupervised learning, an algorithm receives input data without target outputs. The algorithm can use the input patterns to compute clusters of inputs that seem to belong to each other, or it can derive particular features or frequently occurring sub-patterns from the data.

Whereas the aforementioned algorithms work with a dataset of examples to generate a model, reinforcement learning (RL) algorithms train an agent, which is a software program that can select and execute actions based on its inputs. Furthermore, such RL agents interact with an environment through which it receives experiences. Therefore, RL methods do not need a dataset, but are connected to a simulator or the real world. The experiences obtained by an RL agent are generally its observations of the environment, the actions which it executes, and rewards which it receives from selecting actions in particular environmental situations. The goal of the RL agent is to maximize the sum of rewards that it obtains over time.

In particular problems, such as learning to optimally play a game, the reward is given at the end of the game, and is simply 1 for winning, -1 for losing, and 0 for a draw. Even though the agent only receives such *sparse* rewards, it can still learn to play the game very well in many cases. Currently, the most impressive demonstration of the power of RL is Google DeepMind's AlphaGo Zero system [47] that learned to play the complex game of Go extremely well by only playing games against itself. AlphaGo Zero was able to beat its predecessor AlphaGo [45], which first learned from games played by human players and won a match against the human Grandmaster Lee Sedol in 2016 with $4{-}1$.

In this paper, first different kinds of sequential decision making problems for RL are examined. Then we will study several RL algorithms that learn from the experiences obtained by the agent in order to maximize its performance over time. We then examine the issue of exploration, which enables an agent to explore the results of selecting different actions, and which is necessary to learn to perform optimally. To deal with very large state spaces, function approximation techniques are necessary and the most often used techniques will be explained. Then some other topics in RL will be covered, which increase the speed or applicability of RL systems. After having explained the ingredients of RL systems, a number of different applications will be finally described for which RL can be used effectively.

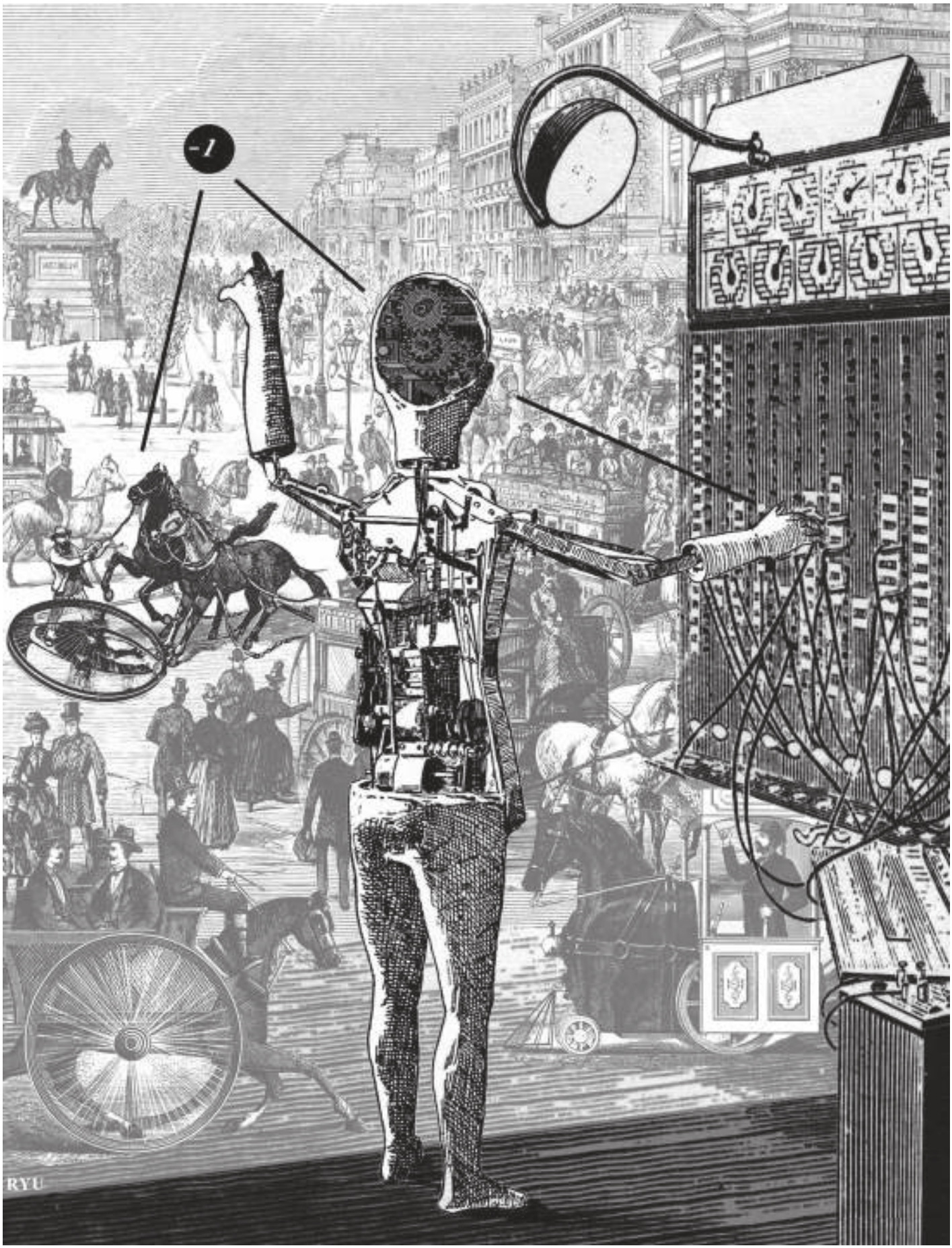


Illustration: Ryu Tajiri

Sequential decision making

Reinforcement learning algorithms enable an agent to optimize its behavior by learning from the interaction with the environment. At each time step t, the agent perceives the state s_t, and uses the state information to select and execute action a_t. Based on this executed action, the agent transits to a new state s_{t+1} and receives a scalar reward r_t. The interaction history of the agent with its environment is $h_t = s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_t$. The interaction between the agent and its environment is illustrated in Figure 1. For some problems, the agent cannot perceive the full state information, and only receives a partial observation o_t instead of s_t. An important property of a sequential decision making problem is whether it has the Markov property, which holds if the current state s_t and action a_t contain sufficient information to predict the next state and the expected received reward: $P(s_{t+1}, r_t | s_t, a_t) = P(s_{t+1}, r_t | h_t, a_t)$. This means that instead of needing to use the whole interaction history h_t, only the current state s_t can be used to select the optimal action. This simplifies matters a lot, since the history could be a very long sequence of previous states and previously executed actions.

There are different kinds of sequential decision making problems. The simplest one is a finite Markov Decision Process (MDP) [58], which has the Markov property and is defined by:

- A set of states S, where $s_t \in S$ denotes the state at time t.
- A set of actions A, where $a_t \in A$ denotes the action selected at time t.
- A transition function $T(s,a,s')$, which specifies the probability of moving to state s' after selecting action a in state s.
- A reward function $R(s,a)$, which sends a reward signal to the agent for executing action a in state s. r_t denotes the reward obtained at time step t. For simplicity we denote the average reward obtained by executing action a in state s also by $R(s,a)$.
- A discount factor γ that makes rewards received further in the future less important than immediate rewards, where $0 \leq \gamma \leq 1$.

The goal in an MDP is to learn a policy, $\pi(s) \rightarrow a$, mapping states to actions in such a way that the agent receives the highest cumulative discounted reward sum in its future. This cumulative discounted reward sum is called the *return* and is defined as:

$$R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i} = r_t + \gamma R_{t+1}.$$

Note that an infinite horizon (future) is used here. By having a discount factor $\gamma < 1$, it is guaranteed that the sum is bounded. Instead of directly optimizing the action-selection policy, value-function based RL algorithms use state-value functions or state-action value functions. The state-action value function $Q^\pi(s,a)$ denotes the expected sum of discounted rewards obtained when the agent selects action a in state s and follows policy π afterwards:

$$Q^\pi(s,a) = E\left(\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right).$$

Where E denotes the expectancy operator. For small-sized finite MDPs, the Q-function can be stored in a lookup table of size $|S||A|$. The goal is then to learn the Q-function Q^* of the optimal policy, for which the following holds:

$$Q^*(s,a) \geq Q^\pi(s,a) \quad \forall s,a,\pi.$$

When the optimal Q-function is known (either computed or learned), the agent can always select the optimal action with the highest Q-value in some state s:

$$\pi^*(s) = \arg\max_a Q^*(s,a).$$

When the MDP is known a priori, it is possible to compute the optimal Q-function with dynamic programming techniques such as value interaction [7]. Bellman noted that the following must hold for the optimal Q-function:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s') \max_b Q^*(s',b).$$

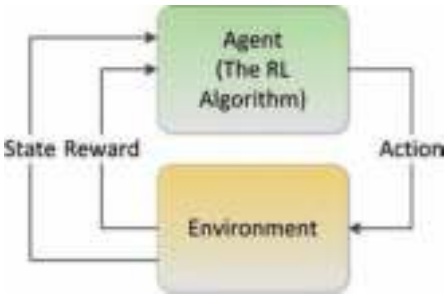


Figure 1 The interaction process between an RL agent and its environment. The agent receives as perception of the environment the state information and uses this to select an action. After this, the agent receives a reward and goes to a next state.

This equation is known as Bellman's optimality equation and uses an efficient recursive formulation. The right-hand side of this equation is basically using a one-step lookahead in the future and could be used to update the Q-value from the left-hand side. The dynamic programming algorithm called value iteration makes use of this to efficiently compute the optimal policy for a known MDP. The algorithm starts with an arbitrary policy π and an arbitrary value function V and repeats the following steps for all $s \in S$. First, the Q-values for all actions in state s are computed:

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s') V(s').$$

Then, the new value function is calculated:

$$V(s) = \max_a Q(s,a).$$

Then, $\pi(s)$ is changed so that the action a which maximizes the Q-function in a state s will be chosen:

$$\pi(s) = \arg\max_a Q(s,a).$$

The algorithm halts when the largest difference between two successive value functions is smaller than ϵ, some positive parameter with a value close to 0. During each iteration, an additional lookahead step in the future is performed. Finally, with a discount factor $\gamma < 1$, the far future is discounted so much, that more iterations do not change the value function anymore.

Although in a lot of research about reinforcement learning, the MDP framework is used, there are also more complex sequential decision making problems. One of these is the Partially Observable Markov Decision Process (POMDP) framework [28]. In a POMDP, the Markov property does not hold, because the agent only receives an observation o_t in each state using an observation function $o_t = O(s_t)$. One problem of POMDPs is that the same observation may be received when the agent is in different states, which is called *perceptual aliasing*. Since the observation does not fully describe the current state, the Markov property does not hold. For this reason, it is necessary to make use of the history h_t instead of only the current observation o_t in order to enable the agent to choose optimal actions.

There exist different algorithms to compute solutions to a priori known POMDPs. The most commonly used approach is to

use the history h_t to create a belief state $b(s_t)$, which is a probability distribution over the entire state space [13]. The probability that the agent is in each state given the history can be efficiently computed using recursive formulas. Then dynamic programming can be used with the belief states in order to compute the optimal policy. The complexity of solving POMDPs is however much larger than that of solving MDPs. Whereas MDPs can be solved in polynomial time, solving POMDPs soon becomes intractable with more states, actions and observations.

Other sequential decision making problems exist when multiple agents learn at the same time. In this case, the Markov property also does not hold anymore, because the actions of other agents change over time and influence the goodness of selecting a particular action in some state for a specific agent. Furthermore, in multi-agent sequential decision making problems, all agents may have a cooperative reward function, but there may also be competitive individual reward functions.

Reinforcement learning algorithms

For most real-world applications a prior model of an MDP (or POMDP) is not known beforehand. Furthermore, the state-action space may be high-dimensional or continuous, so that dynamic programming cannot be effectively used. In that case, RL algorithms can be used to learn the optimal policy by interacting with the environment. The main idea of an RL algorithm is to use each experience of the agent in the form (s_t, a_t, r_t, s_{t+1}) in a way to improve the policy or Q-function. The Q-function can be stored in a lookup table for small state-action spaces, but can also be approximated with function approximators. The most often used algorithm is Q-learning [61,62]. In online Q-learning the Q-value of a state-action pair is updated using the experience by:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1},b) - Q(s_t,a_t)).$$

Where α is the learning rate that determines how much the agent learns from the current experience. In case the last state s_{t+1} is a terminal state, the update rule becomes:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_t - Q(s_t,a_t)).$$

Because most environments are stochastic, it is not always a good idea to use a large learning rate to speed up the learning process. This is because learning on one experience also means forgetting about previous experiences. With a finite number of states and actions, Q-learning converges to the optimal policy when all state-action pairs are visited for an infinite number of times and the learning rate is properly annealed [22]. Note that this means all actions should be tried out infinitely often in each state. Therefore always selecting the action with the highest current Q-value in a state is not good for optimizing the policy. There is also the need to *explore* alternative actions, which currently do not seem optimal, but which could lead to better future states and higher rewards. Exploration in RL will be described in the following section.

Q-learning is called an *off-policy* RL algorithm. Due to the necessity to use an exploration strategy, the behavioral policy that selects actions, is not the same as the greedy policy that is the result of the learning dynamics. Because Q-learning uses the $\max$ operator in its update rule, Q-learning is able to learn the optimal Q-function (and therefore policy) even if always random actions are selected during the learning process. This has several advantages, since it usually allows higher exploration rates than *on-policy* RL algorithms. A well known on-policy RL algorithm is SARSA [39,50] which stands for state-action-reward-state-action. SARSA updates the Q-value of action a_t in state s_t in the following way:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_t + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)).$$

As can be seen from this equation, instead of the best thought action in state s_{t+1}, the actual selected action in the next state is used to update the Q-value. Because SARSA is an on-policy RL algorithm, it is not able to learn an optimal policy as long as there is exploration. Therefore, SARSA only converges to the optimal policy and Q-function if the same conditions hold as for Q-learning and if the exploration policy becomes greedy in the limit of infinite exploration (GLIE) [48].

The difference between the learned behavior when off-policy Q-learning or on-policy SARSA is used can be best illustrated through an example [51]. Given the grid shown in Figure 2, the agent can choose to move North, West, South and East. If an action is not possible (the agent goes outside the grid), the agent remains in the same state. For every action a negative reward of -1 is given except when the agent goes to a state where it falls from the cliff (in yellow), where the agent dies and receives a reward of -100 after which a new epoch is started. When the agent is in the goal state, an epoch terminates as well and the agent does not receive negative rewards anymore. When an epoch ends, the agent starts again in

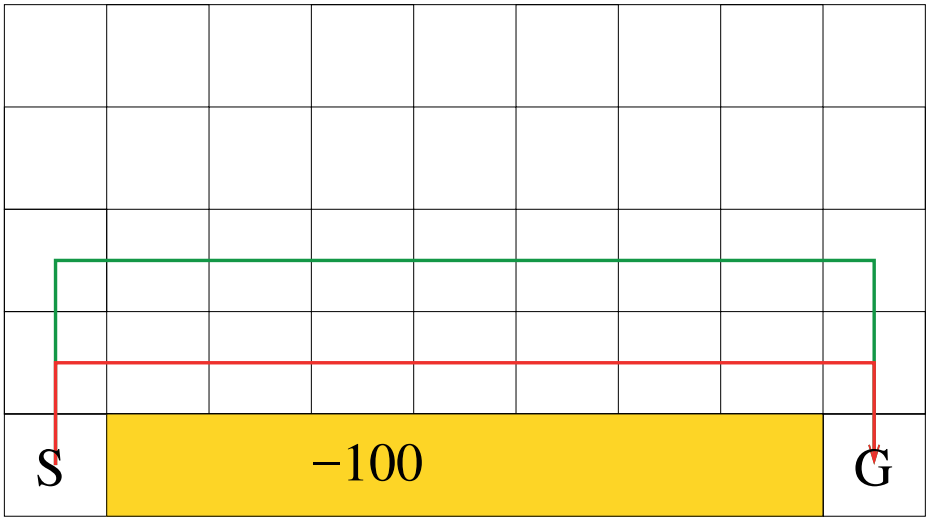


Figure 2 The cliff environment illustrating the difference in learning dynamics between Q-learning and SARSA. The goal is to find the shortest path from the start state S to the goal state G, however a large penalty is given if the agent falls off the cliff. Q-learning will learn the optimal red path, whereas SARSA will converge to a slightly longer path (in green). On the other hand, while learning, the Q-learning agent will fall off the cliff much more often due to exploration actions than the SARSA agent, which takes the possible exploration actions into account when learning the policy.

the start state and this is repeated for a large number of times.

The problem is deterministic, but the agent sometimes chooses an exploration action with some probability. Q-learning will converge to the optimal path shown in red in the figure. SARSA will converge to the green path. It can be seen that the final path of SARSA is longer. This is because the SARSA agent takes into account its exploration actions which often bring it to fall off the cliff when learning. Therefore, the SARSA agent will learn to avoid coming close to the cliff. The result is that the SARSA agent obtains a higher average reward while learning, but after the learning process has converged to a stable policy, the agent receives a slightly lower sum of rewards. Although this is a very simple problem, it clearly shows the differences between off-policy and on-policy RL algorithms.

There are a number of other reinforcement learning algorithms. QV-learning [66] uses two value functions: the state value function $V(s)$ and the state-action or Q-function $Q(s,a)$. The Q-function is trained based on the value function and the value function is trained by using the immediate reward r_t and the value of the next state.

In actor-critic algorithms, the critic uses temporal difference learning [49] to train the value function $V(s)$, and the actor is used to select actions. In the actor-critic framework, the actor is trained using the critic's evaluations. A main advantage of actor-critic algorithms is that it becomes easier to deal with continuous action spaces. If a Q-function $Q(s,a)$ is used in a critic-only system, it is complex to select the continuous action that maximizes the Q-value. With actor-critic approaches, the actor can immediately output the action that should be selected.

A related class of algorithms are policy-gradient algorithms. In their basic form, these algorithms only use an actor with a policy and no critic. One of the earliest approaches, REINFORCE [72], trains the policy using the full (Monte-Carlo) return. If this return is high, then the policy is adjusted so that the taken actions will be strengthened (reinforced). Otherwise, if bad things happen, then the policy is changed to circumvent selecting the same actions.

Another class of algorithms is called evolutionary reinforcement learning [64]. Evolutionary algorithms can be used for many different kinds of problems such as combinatorial optimization problems. They maintain a population of individuals that encode a policy and each individual is tested to obtain its fitness value for the task at hand. Then individuals are selected based on their fitness value and the best individuals are allowed to reproduce most often. New offspring can then be created by mixing the information of the two selected parent individuals using crossover operators and some randomness is added by using a mutation operator. These evolutionary RL methods have been successfully applied to different problems such as playing Othello [33] and were also successfully used to learn policies for Atari games [19]. Value-function based RL and evolutionary RL have also been combined by a number of researchers, see [15] for a review of such combinations.

RL algorithms sometimes need very many experiences to learn a good policy. One of the reasons is that algorithms such as Q-learning and SARSA only update a single Q-value given one new experience. There exist several methods to improve on this. Eligibility traces [49] enable multiple states to be eligible to change their values based on a new experience. Experience replay [27] is a technique in which many experiences are stored in a replay memory and the replay memory is then used to make multiple updates. Experience replay is nowadays often used in many deep RL algorithms [30].

Finally, model-based RL method compute an approximation to the transition and reward functions based on the obtained experiences. When these are modelled, dynamic programming techniques can be used to efficiently compute the Q-function [32, 70]. Although model-based RL techniques are very efficient for finite MDPs with not too many states and actions, it is very complex to use them for high-dimensional and continuous state-action spaces.

Exploration

As mentioned before, an agent needs to make exploration actions in order to be able to learn an optimal policy. However, to get most rewards while interacting with the environment, the agent should also exploit its current knowledge in the form of the Q-values it learned. This leads to the exploration-exploitation dilemma [53]. This dilemma has often been studied using multi-armed bandits (MABs), which are a special kind of MDP with only a single state. In a MAB, the agent can select among k actions, each having its own stochastic reward function $R(a)$. By trying an arm a_t at time step t, the agent receives a reward r_t, and uses this to update its estimate $Q(a_t)$ of the average reward defined by function $R(\cdot)$. When the agent would try out all actions an infinite number of times, then the Q-function $Q(a)$ converges to the average reward $R(a)$ due to the law of large numbers. After this, the agent can simply select the action with the highest Q-value. However, if the agent tries out all actions many times, it also selects suboptimal actions many times, and this is of course not desired.

The MAB-problem has often been used to create proofs for exploration algorithms, as it is simpler than solving a full MDP. Such proofs can for example give guarantees about the maximal amount of experiences needed to learn the optimal action most of the times. Furthermore, exploration algorithms designed for the MAB-problem can also be used as exploration strategy in an MDP. We will now describe a number of often used exploration strategies for solving MDPs [55].

ϵ-greedy exploration is one of the most used exploration strategies. It uses $0 \leq \epsilon \leq 1$ as parameter of exploration to decide which action to perform using $Q(s_t,a)$. The agent chooses the action with the highest Q-value in the current state with probability $1-\epsilon$, and a random action otherwise. A larger value for ϵ means more exploration actions are selected by the agent. Although ϵ-greedy is very simple, it is sometimes hard to beat.

One drawback of ϵ-greedy exploration is that the exploration action is selected uniform randomly from the set of possible actions. Therefore, it is as likely to choose the worst action as it is to choose the second-best action if an exploration action is selected. That is why Boltzmann or softmax exploration [51] uses a Boltzmann distribution to assign a probability $\pi(s_t,a)$ to the actions in order to choose actions with higher Q-values with higher probability:

$$\pi(s_t,a) = \frac{e^{Q(s_t,a)/T}}{\sum_{b=1}^{m} e^{Q(s_t,b)/T}}.$$

$\pi(s_t,a)$ is then the probability with which the

agent selects action a in state s_t. $T \geq 0$ is the temperature parameter used in the Boltzmann distribution. When $T = 0$ the agent does not explore at all, and when $T \rightarrow \infty$ the agent always selects random actions.

Another simple way to allow the agent to explore, is to initialize all Q-values to large values, which is called being optimistic in the face of uncertainty. This makes the agent explore a lot in the beginning during which the values of explored actions will drop. After some time, the initial Q-values have washed out, but the agent has learned which state-action pairs lead to more reward intake than others. A problem of this approach is that the Q-values can drop quite fast when the learning rate is large. Another problem is that this method is difficult to combine with function approximation techniques, which will be discussed in the next section.

The above techniques are all undirected exploration strategies: they only use the Q-values in order to select (exploration) actions. Directed exploration techniques use other information as well. For example, count-based methods [70] keep track of the number of times each action has been selected in each state. This allows the agent to go to areas of the state space, in which it has not (often) been before. Another directed technique is to learn how large the Q-value updates and errors were for particular state-action values. The idea is that large errors correspond to more uncertainty in the actual values and therefore state-action pairs with large errors should be explored more often.

A difficulty of the error-based exploration strategy is that in some states the environment may be very stochastic. In such problems the error will stay high, but these state-action pairs may not be important to visit over and over for learning the optimal policy. Therefore Schmidhuber [41] proposed the idea of curiosity and novelty-based exploration. Here, the agent is attracted to new parts of the state space, but at the same time the agent records if it can really learn new knowledge in these regions of the environment. Think for example about a television. Although black-white noise images are always different, they cannot be predicted and therefore watching these will not be useful when considered by this framework. Curiosity-driven exploration allows the agent to learn to increase its knowledge about the environment, and can be combined with a reward function for solving a task.

Function approximation

In the previous sections, we examined tabular representations for storing the state-action value function. Although these allow to optimally represent the true value function, they cannot be used with very large state spaces or with continuous state spaces. Often decision making problems involve multiple variables representing the state. When this number of variables increases the state spaces explodes, which is called the *curse of dimensionality*. For example, with 20 binary variables representing the state, the number of states is already 2^{20}, or around 1 million. Although current computing power is large enough to cope with 1 million states, things get worse when representing the state of for example all possible chess positions. There are in total around 10^{30} different chess positions, so storing them in a lookup table would be infeasible. To handle this, function approximation techniques can be used to learn to approximate the state-action value function. Next to the decrease in necessary storage space, another advantage of function approximators is that they generalize over the entire state space. If lookup tables are used, states that have not been visited do not have any updated Q-values yet and therefore selecting an action can only occur randomly. With function approximation techniques, similar states will get assigned similar action values and therefore not all states have to be visited in order to select meaningful actions.

In supervised learning, there are many algorithms that can learn mappings from input vectors to real-valued outputs. Basically, all such methods can be used also in reinforcement learning, but in RL research has focused mostly on a specific subset of such methods. The most commonly used function approximators used in RL are: linear function approximation techniques such as tile-coding and radial-basis function networks, multi-layer perceptrons and deep neural networks.

Linear function approximation techniques use a set of basis functions to map the state s_t to an internal representation $\phi_1(s_t), \ldots, \phi_k(s_t)$. To approximate a Q-function these basis function activations are linearly combined:

$$Q(s,a|\theta) = \sum_{i=1}^{k} \phi_i(s)\theta_{i,a}.$$

The advantages of these linear function approximation techniques are that the algorithms are fast, have good convergence properties and are easy to implement. A disadvantage is that the performance of these methods depends heavily on the overall design of the basis functions. One way of designing the basis functions is to use tile coding or CMACS [2, 50]. In tile coding a number of tiles are used with cells inside. Given a state, one cell in each tile is activated, and the activated cells form the new representation of the state. The use of multiple overlapping tilings helps to generalize better for continuous state spaces, as multiple cells which are activated in slightly different input regions can be updated at the same time. A problem of tile coding is that it is difficult to design the tiles and cells for high-dimensional state spaces. In this case, tiles need to combine many different projections of a few state variables, because otherwise there are far too many cells in each tile.

Another linear function approximation technique is the use of radial-basis functions (RBFs). Radial basis functions resemble Gaussian functions without the necessity to be a true probabilistic function that integrates to one. The idea is to place a number of RBFs at specific places in the input space with a specific width, and then given an input state, several of these RBFs are activated based on their distance to the input. The closest RBFs get the highest activations and most of the other RBFs receive an activation of zero. Then the activated RBFs form the new representation of the state. Similarly to the use of tile coding, the use of RBFs has the problem that it is very difficult to fill high dimensional state spaces with these local basis functions. Therefore also this method is best used with not very many state variables.

A more powerful technique to approximate the state-action value function is to use multi-layer perceptrons (MLPs) [38, 63]. Multi-layer perceptrons map an input vector to an output vector using multiple layers of neurons. The neurons are connected with weighted connections, and these weights can be adapted by a learning algorithm to learn the right mapping of input vectors to desired output vectors. One of the first times MLPs were com-

bined with RL, was in Tesauro's backgammon learning program [52]. This program, called TD-Gammon, was able to learn to play backgammon purely by playing games against itself. After around 1.5 million training games, the program was able to attain the level of the strongest human players. MLPs are very powerful in their way to represent non-linear functions, but often need more time and experiences to train and they have much worse convergence guarantees than linear function approximation techniques. In some cases, MLPs combined with Q-learning can even diverge and lead to continuously increasing weight and output values. Still, in many applications MLPs have been fruitfully combined with RL [8, 27, 57].

Inspired by the recent successes in deep learning [25, 42], convolutional neural networks have also been introduced in reinforcement learning, a field that is called *deep reinforcement learning*. In [30], the authors used a convolutional neural network combined with Q-learning to learn to play different Atari games using pixel information as input. The difficulty of using pixels as input is that the input space is incredibly large. However, by using convolutional neural networks, the authors showed that it was possible to play different Atari games at a very good level using only pixel input. In a later paper [31], the authors used the same methodology to learn to play 49 different Atari games at a level comparable to a professional human games tester. Deep RL has also been used with a number of other algorithms such as Monte-Carlo Tree Search [24] to learn to play Go at a level better than any human player [47]. Almost the same system has also been used to learn to play Chess and Shogi at a level much better than any human or previous computer program by letting the system learn from playing games against itself [46]. Nowadays, the field of deep RL attracts a lot of interest from the RL community, and different algorithms have been introduced to make it more efficient [20]. Figure 3 shows how a Deep RL system interacts with the classical Arcade game Breakout to learn to optimize its score.

For solving POMDPs with RL systems, recurrent neural networks can be effectively used. One of the best known and performing recurrent neural networks is the long short term memory (LSTM) network

Figure 3 A Deep RL system learning to play the game Breakout from pixel information as input

[21]. An LSTM can learn to overcome long-time relationships between previous inputs and future predictions and has been successfully used for many time-series applications. Bakker [4] was the first who used the LSTM for solving a POMDP and showed that the LSTM was able to learn to overcome long time lags.

Other topics in RL

We have examined how an agent can use an RL algorithm to learn to optimize its behavior by interacting with an environment. In most research in RL, an agent learns a single state-action value function or policy and uses that to select actions. In the previous decades, a number of new topics in RL have emerged, which are described in detail in [67]. Here, we will have a look at several topics: hierarchical RL, transfer learning, multi-agent RL, and multi-objective RL.

In hierarchical RL a divide and conquer strategy is used to split the overall task in a number of subtasks [5]. In one of the earliest hierarchical RL systems, HQ-learning [69], a higher-level policy is used to learn to select subgoals which the lower-level policy has to attain. This allowed HQ-learning to solve a number of difficult tasks modelled as POMDPs. The options framework [36] introduced the notion of options, which are macro-actions that select multiple actions before finishing. Such options make it easier to learn to solve problems where many actions are needed to arrive at a goal state, since shorter sequences of options can arrive in the goal state. This makes it easier to quicker find the goal and solve the temporal credit assignment problem, which relates to how important a previously executed action was for attaining a goal. MAXQ-learning [14] is another hierarchical RL system in which the overall task is broken up in smaller tasks until at the bottom of the tree primitive actions can be chosen. In general, hierarchical RL systems help to make learning faster when compared to flat learning algorithms.

Humans can learn to solve new tasks faster by using their previous experience with related tasks. This is the field of transfer learning that aims to reuse solutions to previously solved tasks to solve new sequential decision making problems. Transfer learning has been used in different ways such as instance transfer in which experiences are reused, representation transfer in which an abstraction process is used to allow transferring parts of the representation, and parametric transfer to use previous Q-functions for learning to solve new problems [26].

Multi-agent RL is used when multiple agents can learn and interact in the same environment [11]. The reward function can be shared among the agents, leading to cooperative multi-agent systems, or can be individual functions for each agent, possibly leading to competitive systems. One of the oldest multi-agent RL algorithms was a system that used Q-learning on multiple nodes to route packages over a network [9]. The system learned that under busy situations, the shortest route between different parts of the network will become saturated, leading to long waiting times. Therefore the RL system learned to select alternative longer routes if necessary. Multi-agent RL has also been used to learn to find solutions to game theoretical problems such as matrix games. In a matrix game, each agent selects an action and based on all selected actions, each agent receives its own reward. Many matrix games have a competitive nature, and therefore the aim is to learn to converge to a Nash equilibrium, in which no agent is better off when it deviates from the joint action strategy [35].

Although most RL research has focused on scalar reward functions, there also exist RL systems that learn to optimize reward vectors where multiple objectives are assigned rewards at the same time. This is the field of multi-objective RL and its aim is not so much to learn a single policy, but to learn all polices that are not dominated [16, 37, 68, 71]. Policies are non-dominated if they do not obtain a lower cumulative reward on all objectives than another policy. In general it is much harder to learn the set

of non-dominated policies than a single optimal policy, and that is why multi-objective RL has not been used yet for solving very complex problems.

Applications

There are many applications to which reinforcement learning algorithms have been efficiently applied. We will describe a wide range of different applications, starting with game playing, multi-agent problems, and robotics and then discuss applications for medicine and health, electric power grids, and end with personal education systems.

Games provide researchers with a large number of interesting problems having different complexities, but still allow for controlled and repeatable settings. Therefore, they have always played an important part of AI research [73]. The oldest self-learning program that learned to play a game is Samuel's checkers playing program [40]. It combined several machine learning methods and reached a decent amateur level in playing checkers. A very successful attempt to using reinforcement learning to play games is TD-Gammon [52], that learned to play the game of Backgammon at human expert level using temporal difference learning [49] and multi-layer perceptrons. Although in the nineties, learning from 1.5 million games took multiple months, with the current computing power this can be done within several hours. Another board-game that has received a lot of attention from RL researchers is Othello [10, 29, 57, 56]. In [56], structured multi-layer perceptrons were used in which not all board-fields were connected to hidden units, but specific lines or regions were connected to them. This led to much better results and a faster learning process. As mentioned before, AlphaZero obtained an excellent level of playing Chess, but Chess has also been used a long time before to let RL systems to learn to play the game, although the final performance of these RL systems was much worse [6, 54].

Next to using RL for learning to play board-games, RL has also been used for many other types of games. In [8], the authors used Q-learning with a multi-layer perceptron to learn to play the game Ms. Pac-Man. The novel thing was that the system only used seven input units, which extracted higher-level information from the game state. This allowed the system to learn to perform well with only 10,000 training games. Deep RL has also been used for learning to optimize the behavior of Ms. Pac-Man, but using only pixel-information, this took much more training games and led to worse results [31]. Some researchers have decomposed the overall Q-function into a set of single objective reward functions, which are then combined to solve the overall problem. This technique has been successfully used to obtain the highest score with an RL system for the game Ms. Pac-Man in [59].

Many other games such as Starcraft [43], Tron [23], and others have also been used to develop different RL systems that can learn to play them well.

Although games provide researchers with an interesting test problem for their algorithm, the societal relevance is a bit lower than for other problems. In the field of multi-agent reinforcement learning (MARL), different systems have been constructed to learn to optimize the behavior of multiple agents. The network routing problem mentioned earlier is one example. Another successful example was the use of MARL for elevator control [12]. In this system, four elevators have to collaborate to minimize the total waiting time of people wanting to take the elevator from one floor to another. The RL system was able to learn to control the elevators such that the overall waiting time was shorter than with many commonly used techniques. Another MARL problem that has received a lot of attention is traffic light control. Many intersections in cities have traffic lights, but the commonly used controllers can in many cases be improved using RL. In [65], one of the first RL systems is presented that learned to control traffic lights on many intersections. This resulted in much lower waiting times compared to non-adaptive controllers. RL has also been used in a realistic traffic network modelled after the city Tehran where multiple traffic disruptions can take place [3]. Figure 4 shows the used traffic light simulator in [65].

Although RL techniques usually need many interactions with an environment to optimize an agent or controller, RL has also been effectively used for robot control. In [1], first a model was learned from the interaction between an unmanned aerial vehicle (helicopter) and the sky, and then RL was used to learn very complex behaviors such as looping etc. In [60], RL was combined with motor primitives, which take care of particular action sequences, to train a robot to play table tennis. In a recent paper [18], a number of robots was used at the same time to train robots to open a door. There is a growing amount of research using RL for different robot control problems, and of course we cannot cover them all in this short survey.

Therapy planning for patients is another sequential decision making problem where for example different therapies can be giv-

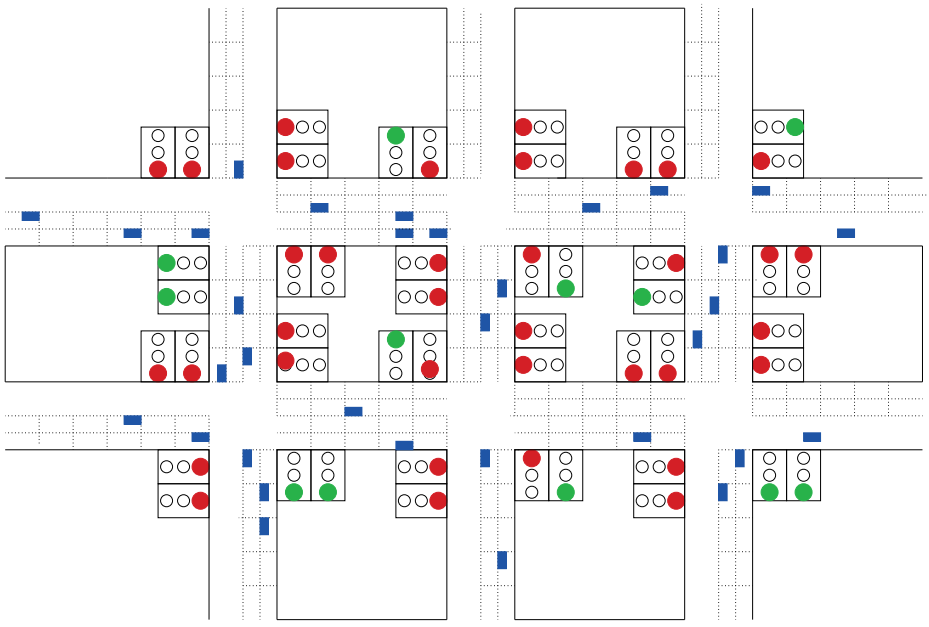


Figure 4 The traffic light simulator consisting of 6 intersections. There are traffic lights for going straight ahead/right or for going left (no collissions).

en to a patient and the goal is to make the patient healthy as soon as possible. Therefore this is also a very good problem for RL, but some things have to be adapted, for example exploration is more complex in this scenario. In [44], an RL system is developed to plan therapies for patients suffering from schizophrenia. In [34], a deep RL system is used for medication dosing. It should be clear that there are many more possible applications of RL to health and medicine.

RL systems have also been used to solve electric power system decision and control problems [17] and make it easier to cope with additional variable power sources such as sun and wind energy. In the finance sector, RL systems can be used to learn from a historical huge amount of data to create automatic trading bots. Also in advertising RL systems have been used to estimate on which advertisement a specific user is most likely to click on. Chatbots have recently been constructed using deep reinforcement learning and because these systems can become better with more interactions and feedback, the end of this is not in sight. Finally, there is some recent work on intelligent tutoring systems that use RL to educate people. These systems can offer much more personalized education than currently used in the classroom.

To conclude, RL systems have been applied to a wide range of different problems. RL systems can continuously learn to improve themselves and do not need a dataset or humans to provide labels to the system. Therefore, reinforcement learning is a very promising direction for Artificial Intelligence to evolve further and to help human kind to cope with many different challenges in life.

References

1 P. Abbeel, A. Coates, M. Quigley and A.Y. Ng, An application of reinforcement learning to aerobatic helicopter flight, in B. Schölkopf, J.C. Platt and T. Hoffman, eds., *Advances in Neural Information Processing Systems* 19, MIT Press, 2007, pp. 1–8.

2 J.S. Albus, A new approach to manipulator control: The cerebellar model articulation controller (CMAC), *Journal of Dynamic Systems, Measurement and Control* 97 (1975), 220–227.

3 M. Aslani, M. Mesgari and M. Wiering, Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events, *Transportation Research Part C: Emerging Technologies* 85 (2017), 732–751.

4 B. Bakker, Reinforcement learning with long short-term memory, in T.G. Dietterich, S. Becker and Z. Ghahramani, eds., *Advances in Neural Information Processing Systems* 14, MIT Press, 2002, pp. 1475–1482.

5 A. Barto and S. Mahadevan, Recent advances in hierarchical reinforcement learning, *Discrete Event Dynamic Systems* 13 (2003), 341–379.

6 J. Baxter, A. Tridgell and L. Weaver, Learning to play chess using temporal differences, *Machine Learning* 40(3) (2000), 243–263.

7 R. Bellman, A markovian decision process, *Indiana Univ. Math. J.* 6(4) (1957), 679–684.

8 L. Bom, R. Henken and M. Wiering, Reinforcement learning to train Ms. Pac-Man using higher-order action-relative inputs, in *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2013, pp. 156–163.

9 J.A. Boyan and M.L. Littman, Packet routing in dynamically changing networks: A reinforcement learning approach, in *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, Morgan Kaufmann Publishers, 1993, pp. 671–678.

10 M. Buro, The evolution of strong Othello programs, *Entertainment Computing*, Springer, 2003, pp. 81–88.

11 L. Busoniu, R. Babuska and B.D. Schutter, Multi-agent reinforcement learning: A survey, *2006 9th International Conference on Control, Automation, Robotics and Vision*. 2006.

12 R. Crites and A. Barto, Improving elevator performance using reinforcement learning, in D. Touretzky, M. Mozer and M. Hasselmo, eds., *Advances in Neural Information Processing Systems* 8, MIT Press, 1996, pp. 1017–1023.

13 B. D'Ambrosio, POMDP learning using qualitative belief spaces, Technical report, Oregon State University, Corvallis, 1989.

14 T. Dietterich, Hierarchical reinforcement learning with the MAXQ value function decomposition, Technical report, Oregon State University, 1997.

15 M.M. Drugan, Synergies between evolutionary algorithms and reinforcement learning, in *Genetic and Evolutionary Computation Conference, GECCO 2015*, pp. 723–740.

16 M.M. Drugan and A. Nowé, Designing multi-objective multi-armed bandits algorithms: A study, in *The 2013 International Joint Conference on Neural Networks, IJCNN*, 2013.

17 M. Glavic, R. Fonteneau and D. Ernst, Reinforcement learning for electric power system decision and control: Past considerations and perspectives, 20th IFAC World Congress, *IFAC-PapersOnLine* 50(1) (2017), 6918–6927.

18 S. Gu, E. Holly, T. Lillicrap and S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in *Proceedings 2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017.

19 M. Hausknecht, J. Lehman, R. Miikkulainen and P. Stone, A neuroevolution approach to general Atari game playing, *IEEE Transactions on Computational Intelligence and AI in Games*, 2013.

20 M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M.G. Azar and D. Silver, Rainbow: Combining improvements in deep reinforcement learning, in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

21 S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Computation*, 9(8) (1997), 1735–1780.

22 T. Jaakkola, M.I. Jordan and S.P. Singh, On the convergence of stochastic iterative dynamic programming algorithms, *Neural Computation*, 6 (1994), 1185–1201.

23 S. Knegt, M. Drugan and M. Wiering, Opponent modelling in the game of Tron using reinforcement learning, in *ICAART 2018: 10th International Conference on Agents and Artificial Intelligence*, 2018, pp. 29–40.

24 L. Kocsis and C. Szepesvári, Bandit based Monte-Carlo planning, in J. Fürnkranz, T. Scheffer and M. Spiliopoulou, eds., *Machine Learning: ECML 2006*, Springer, 2006, pp. 282–293.

25 A. Krizhevsky, I. Sutskever and G.E. Hinton, Imagenet classification with deep convolutional neural networks, in F. Pereira, C. Burges, L. Bottou and K. Weinberger, eds., *Advances in Neural Information Processing Systems* 25, 2012, pp. 1097–1105.

26 A. Lazaric, Transfer in reinforcement learning: A framework and a survey, in M. Wiering and M. van Otterlo, eds., *Reinforcement*

Learning: State-of-the-Art, Springer, 2012, pp. 143–173.

27 L.J. Lin, *Reinforcement Learning for Robots Using Neural Networks*, PhD thesis, Carnegie Mellon University, Pittsburgh, 1993.

28 M.L. Littman, *Algorithms for Sequential Decision Making*, PhD thesis, Brown University, 1996.

29 S. Lucas and T. Runarsson, Temporal difference learning versus co-evolution for acquiring Othello position evaluation, in *Computational Intelligence and Games, 2006 IEEE Symposium*, 2006, pp. 52–59.

30 V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, Playing atari with deep reinforcement learning, arXiv:1312.5602, 2013.

31 V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, *Nature* 518(7540) (2015), 529–533.

32 A.W. Moore and C.G. Atkeson, Prioritized sweeping: Reinforcement learning with less data and less time, *Machine Learning* 13 (1993), 103–130.

33 D.E. Moriarty and R. Miikkulainen, Discovering complex Othello strategies through evolutionary neural networks, *Connection Science* 7(3) (1995), 195–209.

34 S. Nemati, M.M. Ghassemi and G.D. Clifford, Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach, In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2016, pp. 2978–2981.

35 A. Nowé, P. Vrancz and Y.M. De Hauwere, Game theory and multi-agent reinforcement learning, in M. Wiering and M. van Otterlo, eds., *Reinforcement Learning: State-of-the-Art*, Springer, 2012, pp. 441–470.

36 D. Precup and R. Sutton, Theoretical results on reinforcement learning with temporally abstract options, in *Proceedings of the Tenth European Conference on Machine Learning (ECML'98)*, 1998.

37 D.M. Roijers, P. Vamplew, S. Whiteson and R. Dazeley, A survey of multi-objective sequential decision-making, *Journal of Artificial Intelligence Research* 48 (2013), 67–113.

38 D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning internal representations by error propagation, in *Parallel Distributed Processing*, Vol. 1, MIT Press, 1986, pp. 318–362.

39 G.A. Rummery and M. Niranjan, On-line Q-learning using connectionist systems, CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.

40 A.L. Samuel, Some studies in machine learning using the game of checkers, *IBM Journal on Research and Development* 3 (1959), 210–229.

41 J. Schmidhuber, Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts, *Connection Science* 18(2) (2006), 173–187.

42 J. Schmidhuber, Deep learning in neural networks: An overview, *Neural Networks* 61 (2015), 85–117.

43 A. Shantia, E. Begue and M. Wiering, Connectionist reinforcement learning for intelligent unit micro management in Starcraft, in *The 2011 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2011, pp. 1794–1801.

44 S.M. Shortreed, E.B. Laber, D.J. Lizotte, T.S. Stroup, J. Pineau and S.A. Murphy, Informing sequential clinical decision-making through reinforcement learning: an empirical study, *Machine Learning* 84(1-2) (2011), 109–136.

45 D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel and D. Hassabis, Mastering the game of Go with deep neural networks and tree search, *Nature* 529(7587) (2016), 484–489.

46 D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan and D. Hassabis, Mastering chess and shogi by self-play with a general reinforcement learning algorithm, arXiv:1712.01815, 2017.

47 D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel and D. Hassabis, Mastering the game of go without human knowledge, *Nature* 550 (2017), 354.

48 S. Singh, T. Jaakkola, M. Littman and C. Szepesvari, Convergence results for single-step on-policy reinforcement-learning algorithms, *Machine Learning* 38(3) (2000), 287–308.

49 R.S. Sutton, Learning to predict by the methods of temporal differences, *Machine Learning* 3(1) (1988), 9–44.

50 R.S. Sutton, Generalization in reinforcement learning: Successful examples using sparse coarse coding, in D.S. Touretzky, M.C. Mozer and M.E. Hasselmo, eds., *Advances in Neural Information Processing Systems* 8, MIT Press, 1996, pp. 1038–1045.

51 R.S. Sutton and A.G. Barto, *Introduction to Reinforcement Learning*, MIT Press, 1998, 1st edition.

52 G. Tesauro, Temporal difference learning and TD-gammon, *Commun. ACM* 38(3) (1995), 58–68.

53 S. Thrun, Efficient exploration in reinforcement learning, Technical Report CMU-CS-92-102, Carnegie-Mellon University, 1992.

54 S. Thrun, Learning to play the game of chess, *Advances in Neural Information Processing Systems* 7 (1995), 1069–1076.

55 A.D. Tijsma, M.M. Drugan and M.A. Wiering, Comparing exploration strategies for Q-learning in random stochastic mazes, in *2016 IEEE Symposium Series on Computational Intelligence*, *SSCI*, 2016.

56 S. van den Dries and M.A. Wiering, Neural-fitted TD-leaf learning for playing Othello with structured neural networks, *IEEE Transactions on Neural Networks and Learning Systems* 23(11) (2012), 1701–1713.

57 M. van der Ree and M.A. Wiering, Reinforcement learning in the game of Othello: Learning against a fixed opponent and learning from self-play, in *2013 IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL)*, 2013, pp. 108–115.

58 M. van Otterlo and M. Wiering, Reinforcement learning and Markov decision processes, in M. Wiering and M. van Otterlo, eds., *Reinforcement Learning: State-of-the-Art*, Springer, 2012, pp. 3–42.

59 H. van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes and J. Tsang, Hybrid reward architecture for reinforcement learning, arXiv: 1706.04208, 2017.

60 Z. Wang, A. Boularias, K. Muelling, B. Schoelkopf and J. Peters, Anticipatory action selection for human-robot table tennis, *Artificial Intelligence* 247 (2017), pp. 399–414.

61 C.J. Watkins and P. Dayan, Q-learning, *Machine Learning* 8(3) (1992), 279–292.

62 C.J.C.H. Watkins, *Learning from Delayed Rewards*, PhD thesis, King's College, Cambridge, UK, 1989.

63 P.J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, PhD thesis, Harvard University, 1974.

64 S. Whiteson, Evolutionary computation for reinforcement learning, in M. Wiering and M. van Otterlo, eds., Reinforcement Learning: State-of-the-Art, Springer, 2012, pp. 325–355.

65 M. Wiering, Multi-agent reinforcement learning for traffic light control, in *17th International Conference on Machine Learning*, 2000, pp. 1151–1158.

66 M. Wiering, QV(lambda)-learning: A new on-policy reinforcement learning algorithm, in D. Leone, ed., *Proceedings of the 7th European Workshop on Reinforcement Learning*, 2005, pp. 29–30.

67 M. Wiering and M. van Otterlo, *Reinforcement Learning: State of the Art*, Springer, 2012.

68 M.A. Wiering and E.D.D. Jong, Computing optimal stationary policies for multi-objective Markov decision processes, in *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, 2007 (ADPRL 2007)*, IEEE, 2007, pp. 158–165.

69 M.A. Wiering and J.H. Schmidhuber, HQ-learning, *Adaptive Behavior* 6(2) (1997), 219–246.

70 M.A. Wiering and J.H. Schmidhuber, Efficient model-based exploration, in J.A. Meyer and S.W. Wilson, eds., *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior: From Animals to Animats* 6, MIT Press/Bradford Books, 1998, pp. 223–228.

71 M.A. Wiering, M. Withagen and M.M. Drugan, Model-based multi-objective reinforcement learning, in *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL*, 2014.

72 R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Machine Learning* 8 (1992), 229–256.

73 G.N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*, Springer, 2018.

Michael Repplinger
ILLC
University of Amsterdam
mpjrepplinger@gmail.com

Lisa Beinborn
Language Technology Lab
University of Duisburg-Essen, Germany
lisa.beinborn@uni-due.de

Willem Zuidema
ILLC
University of Amsterdam
zuidema@uva.nl

Vector-space models of words and sentences

How can we compute with words? Natural Language Processing is a research field focused on developing mathematical and computational models of language. For decades, models in this field were using techniques from discrete mathematics, but in recent years — with the rise of 'deep learning' — words and sentences are increasingly modelled with continuously valued numerical vectors. Can these models deal with the endless creativity of language, where ten thousands of words can be combined into an unbounded number of potential sentences? Michael Repplinger, Lisa Beinborn, Willem Zuidema discuss the four steps the field has gone through to arrive at the current state-of-the-art vector-space models of sentences.

Language, in written, spoken or signed form, is all around us. Most of our daily communication makes use of language, most of what we learn in school is conveyed through language, and most of the knowledge that humanity has built up is passed on to future generations through language. For allowing computers to take part in daily interactions with humans and to make use of the accumulated knowledge of humanity, we need mathematical models of language — models that allow computers to translate the noisy spoken, written or signed forms into internal representations with which they can compute.

The design of mathematical models for many different aspects of language and speech has a long history going back to at least the Indian grammarian Panini (6th, 5th or 4th century B.C.) and the Greek philosopher Aristotle (4th century B.C.). From the early 20th century, models based on formal logic became popular. Linguists like to point out that language comes so natural to us that we are largely unaware of its complexity; logical models played a key role in uncovering and detailing the complexity of language structure.

Although logic-based models continue to be important in linguistics, in Natural Language Processing, the field that studies language technology on computers — they are in the last few years rapidly making way for so-called 'vector-space models'. In those models, words are represented as numerical vectors, and sentence meanings are computed using a variety of operations from linear algebra.

In this article, we describe these developments, going over four major steps in the development of mathematical models of meaning, leading-up to current vector-space models of sentence meaning, compatible with sophisticated techniques from the deep learning field.

Step 1: Montague semantics — or: celebrating the sentence, ignoring the word

The American logician Richard Montague (1930–1971) is widely credited with providing the first successful attempt to formalize the semantics of a substantial 'fragment' of natural language. He developed what we now call *Montague Semantics* or *Montague Grammar* in a series of seminal papers [28, 29, 30].

Montague Semantics provided a systematic way to translate natural language to a logical language. For instance, when processing the sentence 'Gottlob loves Yoshua', the goal is to end up with the logical expression $\mathrm{love}(\mathrm{g})(\mathrm{y})$, where love is a binary predicate, describing a relation between the constants g (Gottlob) and y (Yoshua).

To achieve this translation, Montague proposed an ingenious system that combined insights from various modeling traditions in linguistics and logic. From categorical grammar, he borrowed a system to assign syntactic categories to words. 'Gottlob' and 'Yoshua' are proper nouns, and

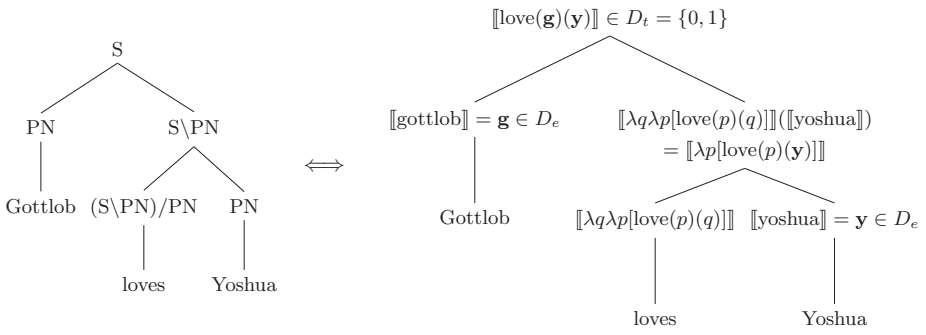


Figure 1 Translation to logical language.

receive the atomic category PN. 'loves' is a so-called transitive verb, that needs an argument on its left (the person that loves) and an argument on its right (the person that is loved). It therefore receives a 'complex category' that contains slashes that indicate what it can be combined with on the left and the right; in the case of love, the category is $(\mathrm{S}\backslash\mathrm{PN})/\mathrm{PN}$. According to the rules of categorical grammar, loves can then first be combined with Yoshua, and again be combined with Gottlob. This process yields a syntactic analysis depicted in Figure 1 (left).

From logic, Montague borrowed *intensional logic*, a higher-order typed logic (our examples will only use first-order predicate logic). Crucially, however, Montague proposed that the semantic, logical representation is derived simultaneously with the grammatical derivation. To achieve this, he enriched the logical expressions with elements from the *lambda calculus*. Gottlob and Yoshua again get assigned an atomic symbol, g and y respectively. The meaning of 'loves' is represented as $\lambda q \lambda p[\mathrm{love}(p)\,(q)]$. According to the rules of the lambda calculus, the semantics of the first argument that loves get combined with y, ends up in the place of the variable marked with the first λ in the expression q. This semantic derivation is depicted in Figure 1 (right).

On this basis, Montague and others have built an enormous body of work to analyze the semantics of natural language sentences. This work has addressed for instance the subtle ways in which 'some' and 'any' differ in sentences like 'some mathematicians proved a theorem' or 'there wasn't any mathematician that proved a theorem'. Interested readers will find [10] or [15] to be thorough introductions to complete semantic systems in the spirit of Montague's proposal.

Strengths and weaknesses

One of the lasting impacts of Montague Semantics has been that it has highlighted an important feature of human language, known as the *principle of compositionality* (already hinted at in the work of Gottlob Frege [9]). This principle is commonly phrased along the lines of: "The meaning of a complex expression is determined by the meanings of its constituents and the rules used to combine them."

Closely related to compositionality is the notion of *recursion*. Recursively defined processes are widely believed to underly the capacity of speakers of a language to build and understand arbitrary expressions of the language. By recursive *syntactic* processing, speakers can, in principle, build an infinite set of complex expressions from a finite set of simple 'building blocks'. By a parallel recursive *semantic* process, speakers are then able to express and understand a potential infinitude of distinct meanings.

Formal semantics in the symbolic tradition excels at modeling compositionality and recursion — deriving algorithmically the meaning of a sentence from its smaller constituents. This focus on the structural aspects of language comes at a price, however, and the lexical foundation, i.e. the meaning of words, has received much less attention.

Another point concerns the limited integration of individual semantic theories. Montague's original proposal was a 'method of fragments' — identifying a well-delineated syntactic fragment of the language, then formally describing this fragment. However, modern semantic theories generally no longer follow this approach, opting instead for descriptions of individual *semantic phenomena* without confining them to some *syntactic fragment*. Consequently, there is no syntactically defined subset of the language which can then be semantically analyzed in completion. The latter is however a requirement for *computational* models of symbolic semantics [31]. Consequently, the use of these theories for practical computational applications is limited as well.

Step 2: Distributional semantics — or: counting words

More recently, the field of *distributional semantics* emerged, which takes a completely different approach to modeling meaning, focusing on statistical techniques and large-scale modeling of word meaning. Central to distributional semantics is the *distributional hypothesis*, often expressed as: "You shall know a word by the company it keeps" [8]. To turn Firth's slogan into a formal system, we use numerical vectors as representations and fill them with numbers based on counts of how often pairs of words occur near to each other in large databases of text.

Table 1 shows an idealized co-occurrence count, corresponding to a semantic space for the three nouns 'mouse', 'elephant' and 'car'. Values in this table indicate how often a target word, i.e. the word we aim to represent as a vector, appeared near certain other words in a corpus. For example, 'mouse' and 'animal' co-occurred eight times, while 'car' and 'animal' co-occurred only once. Based on these hypothetical counts, we can represent these nouns as vectors in a four-dimensional semantic space, where the vectors consist of raw co-occurrence counts. In practical systems, dimensionality is usually in the order of hundreds or thousands, and raw counts are usually transformed into (weighted) frequency values.

The great advantage of vector representation for the meaning of words, is that we can now use standard mathematical tools that apply to numerical vectors to compute similarity between words. One frequently used similarity metric is cosine. Using *cosine similarity* on pairs of these word vectors, we can calculate their se-

	animal	large	small	USB
mouse	8	2	4	3
elephant	9	5	1	0
car	1	4	3	0

Table 1 Idealized context-counts for 'mouse', 'elephant' and 'car'.

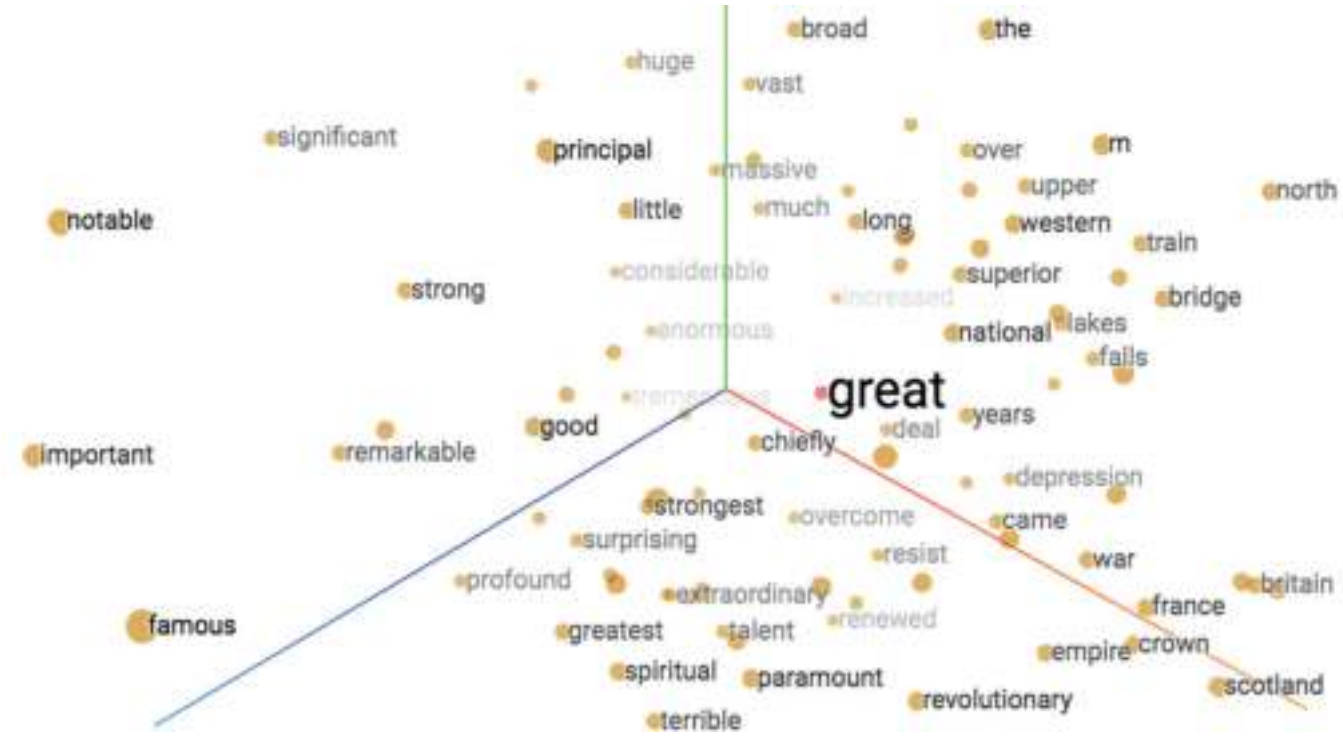


Figure 2 word2vec embedding of *great* and neighboring embeddings (1st/2nd/3rd principal component of 200 dimensional vector embeddings plotted).

mantic similarity: 0.86 (mouse, elephant), 0.57 (mouse, car), 0.61 (elephant, car). This would then represent, as intended, that 'mouse' and 'elephant' are more similar to each other than either of the two is to 'car' (animals versus non-animal), and that 'car' is (slightly) more similar to 'elephant' than to 'mouse' (being somewhat more similar in size). Note also the co-occurrence of 'mouse' with a seemingly unrelated word, 'USB', intended to illustrate the problem of *polysemy*. In most models, the vector representation of 'mouse' would be a 'mix' of contexts where 'mouse' refers to a rodent and contexts where the word refers to a computer component.

Modeling choices

Various modeling choices need to be made in the specification of a semantic vector space model. First, one must define what constitutes *context*, i.e. what we consider 'near to each other'. Commonly, this context is defined as 'the N neighboring words of the target word', where N is the *context window*, another parameter. Other choices are possible however. For example, context can be defined on a sub-word level, e.g. based on single characters. The choice of a similarity metric (Euclidean, correlation, cosine) also influences what it means for different words to be similar in meaning.

Models derive semantic information from the analysis of lexical co-occurrence, i.e. they are based on *context counting*. These counts are usually processed further by *weighting schemes*. Intuitively, these processing steps can be seen as adjusting the raw counts for word frequency, giving more weight to words that are rare but informative. One frequently used method used for this purpose is *pointwise mutual information*.

Another frequently employed processing step is to perform *dimensionality reduction* on the derived representations, for example by non-negative matrix factorization (NMF) or singular value decomposition (SVD) [3]. The dimensionality reduction step is motivated by two main concerns: computational efficiency, and possibly greater *generalization* capacity of the model [20]. The latter effect can result from merging (similar) contexts, i.e. associating a word with contexts of similar words even though it might have never appeared directly in these contexts itself, thus allowing the model to uncover additional similarities.

Step 3: Neural word embeddings — or: Learning to predict words

The classical distributional models described so far learn the meaning of words by analyzing the *counts* of context items, given some target word. Recently, a new class of models has emerged based on neural networks. These models are trained to either predict the most likely word given some context, or, in reverse direction, the most likely context for a given word. The word vectors that emerge as a side-effect of this prediction task, have turned out to be of much higher quality than the word vector from classical distributional semantics.

The origins of neural word embeddings can be traced back to the proposals of Hinton [16] and Bengio et al. [4], who used neural architectures in the derivation of word meanings. The currently most successful embedding algorithms were proposed by Mikolov et al. [23] and Mikolov et al. [22], accompanied by an efficient implementation dubbed `word2vec`.These modern neural word embeddings can be seen as feedforward neural networks (see box 'Feedforward neural networks') without hidden layers and nonlinear activation functions — the latter were identified as computational bottlenecks of the original models.

`word2vec` embeddings are based on two closely related algorithms. The first model, *Continuous Bag-of-Words* (CBOW), learns to *predict a word*, given a context of surrounding words. A 'projection layer'

turns discretely encoded (sparse) word representations into continuous (dense) representations, which are then fed into a matrix — shared for all context words regardless of their position, hence the name 'bag-of-words' — for an output prediction of the most likely middle word for a given context. The second type of models is trained to predict in the opposite direction; given a word, the goal is to maximize the log probability of its surrounding context, i.e. models learn to *predict the context*, given an individual word. The context words do not need to appear consecutively in the corpus, i.e. individual words can be *skipped* when determining the context — referred to by the algorithm's name, *Skip-Gram*.

Linguistic regularities inside embeddings
Neural embeddings gained widespread attention for their representation of complex syntactic and semantic relational information. Very influential was the demonstration by Mikolov et al. [24] that a relation-specific, constant *vector offset* exists between the vector representations of related word pairs. Mikolov reported, for instance, that when you substract the vector for 'man' from the vector for 'king' and then add the vector for 'woman', you end up very near to the vector for 'queen':

$$v_{\text{king}} - v_{\text{man}} + v_{\text{woman}} \approx v_{\text{queen}}.$$

More generally, given a constant offset, the embedding space can be queried for an answer to *analogy questions* of the form: "word 1 is to word 2 as word 3 is to word 4". In this query, words 1, 2, 3 are given, while word 4 must be found in order to answer the question. Expressed as vector space operations, using cosine similarity to measure semantic similarity, the analogy query is defined as:

$$\arg\max_{v_4}(\cos(v_4, v_3 - v_1 + v_2)). \qquad (1)$$

Some of the other results produced by these analogy queries are:

$$v_{\text{Paris}} - v_{\text{France}} + v_{\text{Italy}} \approx v_{\text{Rome}},$$
$$v_{\text{apple}} - v_{\text{apples}} \approx v_{\text{car}} - v_{\text{cars}}.$$

These results have been interpreted as evidence that embeddings contain structure encoding a *gender* relation or feature, can relate countries and their capitals, and can learn a systematic *syntactic* relation between singular and plural word forms.

Arora et al. [1] provide theoretical support in favor of these claims, by showing that embedding algorithms like `word2vec` actively *impose* linear structure on the language data. The authors argue that this linearization effect stems from the *lower-dimensional* model internal representations, and because embedding models are effectively *nonlinear* data processors even though they do not contain the non-linear activation function of a full-power neural network.

Limitations of context-based approaches
Objections against embedding models have frequently been raised, noting that such models operate at a low level of the language (words or characters). Further criticism stems from the fact that models of this approach learn meaning by a comparably 'shallow' context analysis, without explicitly accounting for syntactic or semantic compositionality. Such broad criticism appears to be unwarranted: the examples shown above, of relational structure emerging in embedding models, strongly suggest that distributional models cannot be simply dismissed as 'linguistically insufficient'. Even though their architecture does not explicitly account for compositionality, the models succeed in extracting highly structured information from language data, through a combination of sophisticated statistical machinery, and due to their ability to process enormous amounts of data.

At the same time, some language phenomena pose major challenges for the class of purely context-based models. One example is the meaning derivation of *logical operators*, such as negation. In order to learn (sentential) negation from the context analysis alone, the intended meaning of negation would have to be fully expressed in the context distributions of sentences and their negation. It seems obvious however that human speakers can express (and understand) the negation of a sentence without knowledge of any 'neighboring' sentences. Confirming this intuition, Mohammad et al. show that highly contrasting items (including aspectual negation) tend to occur in very similar contexts. Models that purely learn from context, without the ability to deconstruct it if necessary, are then missing the relevant statistical information that would allow them to derive the correct meaning of negation.

Another obstacle for models that rely entirely on contextual learning is *sparsity of data*. Individual words occur abundantly across different contexts, but the frequency of combinations of words declines exponentially with the length of the sequence. Phrases, and short sentences are still encountered frequently enough to allow learning through contextual analysis alone. Long sentences on the other hand appear only infrequently, or are unique in the worst case, even in large data sets. Context-based learning however generally requires training on a large number of instances of an expression, thus making sentence length a limiting factor for context-only models.

Step 4: Compositional distributional semantics — or: having your cake and eat it
Speakers of a language are able to generalize from previously encountered expressions to expressions that a speaker never heard before. They achieve this by analyzing previously heard utterances as being built up from component parts, and by reusing these parts in all kinds of new combinations.

The symbolic models of language in the tradition of Montague emphasized the compositional semantics of sentences, but largely ignored how the meaning of words can be modelled and moreover relied on hand-built grammars specifying the syntactic and semantic properties of words. The distributional semantics tradition successfully developed vector representations for words. The recent neural word embedding models built on those successes, and moreover showed that automatically learned word representations encode many linguistically relevant relations between words. But distributional and neural word vectors have little to say about how sentence meanings can be constructed.

In the last few years, much research is devoted to bringing together insights about compositionality from the symbolic tradition, and insights from vector-space models of word meaning from the distributional and neural traditions: compositional distributional semantics.

Investigating different types of composition
Two landmark articles from this field of research are Mitchell and Lapata [25, 26]. The authors systematically evaluate different types of composition functions that can be

used in vector space models to compose sentence meaning from the meaning of smaller units, such as words.

Additive composition. The class of models based on *additive* composition is defined as:

$$z = Vx + Wy$$

where V, W are matrices, and composition is given by matrix multiplication. The very simplest instance of this class is given by vector addition:

$$z = x + y.$$

Mitchell and Lapata point out the clear limitations of this approach, such as insensitivity to word order due to commutativity of vector addition. For example, recursive application of vector addition would derive identical meanings for the phrases 'man bites dog' and 'dog bites man'.

Adding scalar weights for each vector results in the *weighted additive* model:

$$z = \alpha x + \beta y.$$

Here, left and right input vectors are scaled (uniformly, for each vector) by parameters α, β which are optimized on a development set.

Employing scalar weights or full matrices fixes the commutativity problem of simple vector addition, but another problem remains: the *blending* of meaning in additive models. Even the most complex (matrix-based) additive models compose vectors through (weighted) sums of their components. Additive composition effectively 'blends' or 'mixes' the meaning aspects of the composed word vectors, which can lead to undesirable results.

Consider for example the phrase 'brown cow', the result of composing vectors for 'brown' and 'cow'. Its intended meaning is a particular type of cow, one that is brown. Ideally, composition would yield a vector that represents the intersection of 'things that are brown' and 'things that are cows'. Recall now that word vectors gather their meaning from co-occurrence counts, and that the 'brown' vector contains meaning elements related to any brown objects encountered in the data: (brown) cows, (brown) dogs, (brown) houses, and so on. Since additive composition is incapable of context-dependent selection of only the relevant meaning aspects, the composed vector for 'brown cow' is bound to include various unrelated meaning aspects contained in the 'brown' vector (e.g. aspects related to brown dogs). It is easy to see that this effect runs counter to the intended meaning of the phrase, as an intersection of the two concepts.

Multiplicative and tensor-based composition. Multiplication of components is suggested as a solution to the *blending* problem above, leading to a general form of *multiplicative* models:

$$z = \mathcal{V}xy$$

where $\mathcal{V}$ is a 3rd-order tensor mapping the two constituent vectors x, y to output vector z. Composition is consequently a bilinear function of the constituents. A simple instance of this class is composition by *element-wise product*:

$$z = x \odot y.$$

The most powerful class of multiplicative models is given by composition with the *tensor product*:

$$z = x \otimes y.$$

Mitchell and Lapata argue that multiplication of components as defined here is a solution to the previous blending problem: Additive composition does not relate the input components directly, adding them independently (at best, scaled by some constant factor) to form the output. Given component multiplication however, values interact directly through their products. For example, if a component with some high numerical value 'interacts' by multiplication with another component that has the value 0, the meaning contribution of the former is limited by its interaction with the latter. The authors describe this form of composition as feature *filtering*, in contrast to the feature *blending* that results from additive composition.

Mitchell and Lapata [25] evaluate the different model classes on a set of semantic similarity tasks, and arrive at somewhat inconclusive results. While the multiplicative models perform well (as predicted by the authors), only the simplest multiplicative model based on point-wise multiplication performs well across different tasks. Models using the tensor product, predicted to be more powerful than other simpler models, perform worse than most other models. Finally, the additive models perform comparably well across tasks, despite the theoretical objections raised against them.

The experimental findings of Mitchell and Lapata [25] are somewhat outdated, due to advances of models in recent years. Their proposed classification, however, had a lasting influence, and is still frequently invoked to classify and relate model architectures.

Modern approaches

Current research on compositional distributional semantics exists in two flavors. One class of models, which we collectively refer to as *type-based tensor approaches*, combines a powerful compositional mechanism with a robust distributional foundation of word meaning — at the cost of very high computational complexity. The approach results from the independent work of two research groups, presented by Baroni and Zamparelli [2] and Coecke et al. [5] approximately in parallel. Informally, these approaches can be seen as a 'translation' of

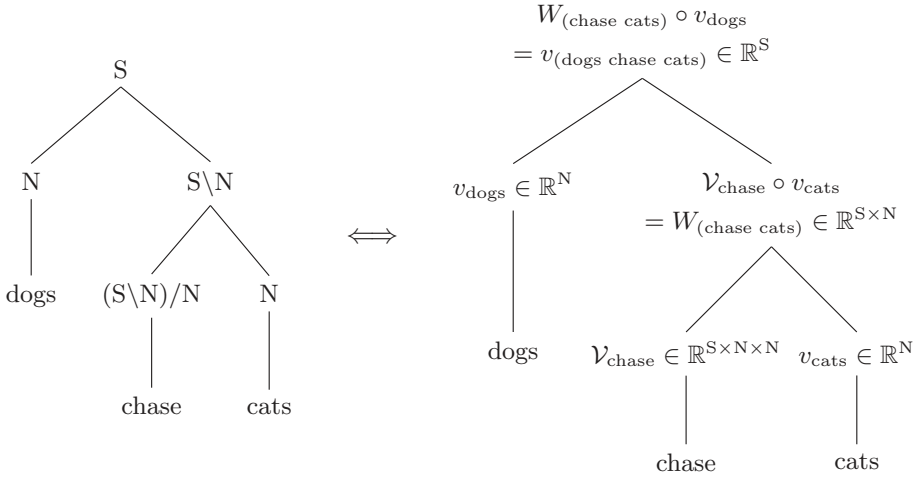


Figure 3 Sentence derivation in type-based tensor models.

the symbolic semantic theories in the Montague tradition into a vector space setting, through the use of higher-order tensors, as illustrated in Figure 3.

The other class of models consists of neural network architectures that — implicitly or explicitly — account for the demands of semantic compositionality. After a period of receiving only little attention, neural network models re-emerged in recent years as powerful, robust models, shown to be capable of solving a plethora of tasks across domains. The new generation of neural models produced outstanding results in the field of computer vision, as well as on language processing tasks such as machine translation, sentiment analysis or information retrieval. Many of these models are presented under the umbrella term of *deep learning*, originally meant to describe neural network models that contain a high number of hidden layers. In the appendix, we go over the main deep learning architectures that have been used to compute sentence representations in recent years.

Conclusions

Compared to mathematics, human language is a hopelessly messy, ambiguous, and redundant system. Yet, language is the carrier of a vast amount of knowledge, and for both scientific and technological reasons there is a need of adequate mathematical models of language.

In developing such models, linguists have looked at many different branches of mathematics. Until a few years ago, most useful tools where found in discrete mathematics: logics to describe the meaning of utterances, grammars of various sorts to describe the structure of sentences, lambda calculus to regulate to combinations of bits of meaning into larger wholes. In recent years, the field of Natural Language Processing is turning to continuous mathematics. Suddenly, words are modelled as continuously-valued numerical vectors, sentences as summations, mutiplications or more complicated combinations of these word vectors. And, importantly, these word and sentence vectors are computed using neural networks, optimized using stochastic gradient decent and other tools from the increasingly rich toolbox of 'deep learning'.

As always when scientific fields go through a paradigm shift, much of the excellent work done in the old paradigm is ignored or discarded. Fortunately, however, researchers in the domain of compositional distributional semantics are finding ways to integrate the main insights from the symbolic and neural traditions.

Acknowledgments

WZ is supported by a Gravitation grant, nr 024.001.006, from the Netherlands Organization for Scientific Research (NWO) to the Language in Interaction consortium. This paper is largely based on chapters 2, 3 and 4 of Repplinger [33]. We thank Dieuwke Hupkes for useful comments.

Feedforward neural networks

A simple feedforward neural network with two hidden layers, $\varphi^{\text{nn2}}(x)$, can be compactly represented in vectorized notation as follows:

$$\begin{aligned} \varphi^{\text{nn2}}(x) &= z, \\ h^1 &= f(W^1 x + b^1), \\ h^2 &= f(W^2 h^1 + b^2), \\ z &= W^3 h^2. \end{aligned} \tag{2}$$

Vector $\mathbf{z}$ is the network output, for an input of vector $\mathbf{x}$. Matrices $\mathbf{W^1}$, $\mathbf{W^2}$ and (bias) vectors $\mathbf{b^1}$, $\mathbf{b^2}$ are the trainable parameters of the network, together computing a linear transformation of the input $\mathbf{x}$. Using a non-linear (activation) function in place of f, the network will however learn complex functions that go well beyond simple linear transformations. Theoretical results by Cybenko [6] and Hornik et al. [8] established that neural networks are in principle able to approximate any function of practical interest to an arbitrary degree of precision, given a sufficiently high number of adjustable model parameters. Figure 4 is an equivalent representation of the network $\varphi^{\text{nn2}}(x)$, in the traditional form of connected *neurons* computing a weighted sum of their input.

Simple feedforward networks are however ill-equipped to process sequences of arbitrary length, which is a basic requirement for the semantic modeling of sentences. Any sentence of a given length can be processed by a feedforward neural network with appropriate input layer dimensions. However, since the input layer dimension is fixed, the same model instance cannot be applied to sentences of any *other* length, preventing models from shared learning across sentences of different lengths. In order to process sequences of arbitrary length, such as sentences, the model should allow for the *recursive* processing of input sequences.

Recurrent neural networks

A simple, but powerful architecture satisfying the requirement of recursive input processing is the *recurrent neural network* (RNN), originally proposed in Elman [7].

The function learned by an RNN model is defined as a recursion over an input sequence of vectors $x_1, \ldots, x_n$. This sequence of input vectors can be chosen quite generally, for example, as vectors representing the *words* of a sentence, or, breaking input down further, as the *characters* of a sentence.

At input step x_i of input sequence $\mathbf{x}$, the output z_i of an RNN is given by:

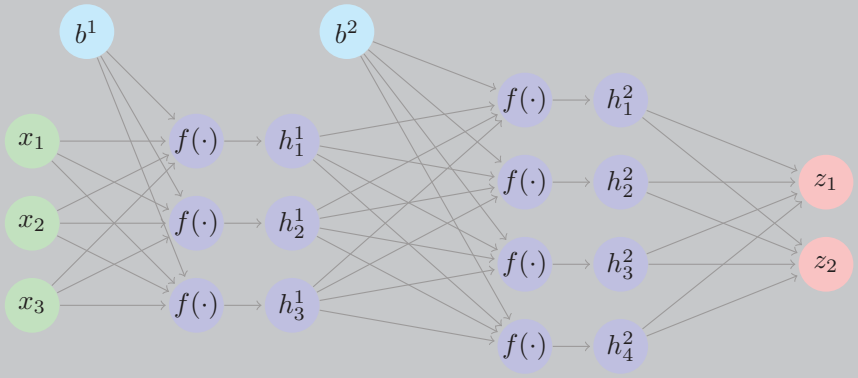


Figure 4 Simple feedforward neural network with two hidden layers.

$$z_i = f(W^{hh} z_{i-1} + W^{hx} x_i + b^h) \quad (3)$$

where $z_i \in \mathbb{R}^H$, and H is the dimension of the hidden layer, $x_i \in \mathbb{R}^X$, $W^{hh} \in \mathbb{R}^{H \times H}$ is the matrix of hidden-to-hidden connections, $W^{hx} \in \mathbb{R}^{H \times X}$ the matrix of input-to-hidden connections, $b^h \in \mathbb{R}^H$ is a bias vector, and f an element-wise nonlinear function, called the *nonlinearity* of a layer. Frequent choices for this nonlinearity are the sigmoid function, $\tanh$, or a rectified linear unit (ReLU).

For finite input sequences, the general recursive definition can be replaced by an *unfolded* version of the model instance. Given an input sequence of length n, the network can be seen as a chain of $n+1$ hidden layers or states, where each state has two outgoing connections (one to the next hidden layer, one being the output at the current step), and two incoming connections (one being the output of the previous hidden layer, one for the input at the current step). The i-th state of this chain is the model representation of the input sequence up to and including input element i, given by summing the linear combination of the i-th input vector, the linear combination of the $(i-1)$-th state or output, and the bias vector, then applying the element-wise nonlinearity f to the resulting vector.

Using simple RNN models, the work of Mikolov [21] constitutes an early, influential exploration of RNNs applied to language tasks. In general, however, most notable results produced by RNN architectures in recent years were in fact *extensions* of the architecture, often produced by the highly successful class of LSTM models, discussed later on.

Recursive neural networks

A structural extension of the basic RNN architecture is the *recursive neural network*, or tree-shaped recurrent neural network (tRNN). The current tRNN architecture was introduced by Socher et al. [33] and was based on earlier proposals of Pollack [32] and Goller and Küchler [12]. The tRNN can be seen as a generalization of RNNs in terms of input structure, in the following sense: While the input sequence of a simple RNN is unstructured, and composition invariably proceeds in one direction, the tRNN architecture allows for the compositional process to be structured by syntactic analysis of the input. In practice, this syntactic analysis is usually provided externally, by providing the model with a parse tree for a given input sentence.

The decision to provide the neural network with a parse tree of the input is motivated by linguistic considerations, attempting to include some of the structural information that lends power to the symbolic models of formal semantics. While the tRNN class of models initially proved to be successful, the field has largely moved on to models that do not require external information (such as parse trees), allowing for much larger data sets to be used in training.

Neural networks with long-term memory

A major challenge when training *deep* neural networks — networks consisting of many stacked hidden layers — is the *vanishing gradient problem*. The cause for this problem relates to the training algorithm of networks, which passes information (gradients) down the network as a chain of *products*. Since individual terms of this chain are often small, their product tends to decrease with the length of the chain, to the point of vanishing. As a result, lower layers of a deep network only receive a greatly diminished learning signal, thus negatively affecting learning success.

LSTM networks

The seminal work of Hochreiter and Schmidhuber [17] presented a solution to the problem, by introducing the *long short-term memory architecture* (LSTM). We only describe the general idea behind the approach here, and refer the reader to Graves [13] for a technical explanation of the mechanisms.

The LSTM architecture, based on the (simple) RNN model of (3), adds *memory cells* and *(control) gates*, allowing a higher degree of retainment of gradient information across network layers. Memory cells are vectors retaining past gradient information, where access to these cells is controlled by three types of gates (*input*, *output*, and *forget* gates). Intuitively, these gates can be seen as vector space versions of logic gates, interacting with the components of the memory cells by pointwise multiplication with values near 0 or 1, i.e. *soft* boolean values. During training, stored gradient information and the gates interact to *preserve* old and *select* new gradient information that will be passed downwards in the network. This mechanism leads to major improvements in training effectiveness of deep networks, and most major results in recent years by models of the RNN class were produced by LSTMs, or further model extensions of the RNN architecture, with added LSTM gates.

RNN models using LSTM gates generated major results on several language tasks. In an early study of LSTMs and language processing, Gers and Schmidhuber [11] showed that their model can learn simple context-free and context-sensitive languages, e.g. strings of the form $\mathrm{a}^n\,\mathrm{b}^n$ and $\mathrm{a}^n\mathrm{b}^n\mathrm{c}^n$, respectively. In Graves [14], LSTM models are used to generate novel sentences after being trained on Wikipedia data, and learn to produce sentences in realistic script (i.e. the model learned *handwriting*).

Tree-structure information for free?

As mentioned above, recursive neural networks, i.e. tree-shaped RNNs, make use of explicit syntactic information to guide the processing of sentences. LSTM models have been suggested as effectively replacing the need for such explicit syntactic information, due to their ability to store (training) information across the processing of long input sequences like sentences. While syntactic information is given to the tree-structured networks *explicitly*, LSTM models possibly can rely on *implicit* syntactic information through their storage mechanism. Whether syntactically guided processing is a useful or necessary feature of distributional models is not conclusively answered yet. It should be noted however that the two architectures can be combined, i.e. tree-structured networks can be enriched by adding a memory mechanism. See, for example, the proposals of Le and Zuidema [19] and Tai et al. [34], extending tRNN architectures with LSTM gates to improve training efficiency of deep networks, and help with modeling long distance dependencies.

References

1 Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma and Andrej Risteski, Rand-walk: A latent variable model approach to word embeddings, *Transactions of the Association for Computational Linguistics* 4 (2016), 385–399.

2 Marco Baroni and Roberto Zamparelli, Nouns are vectors, adjectives are matrices: Representing adjective noun constructions in semantic space, in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2010, pp. 1183–1193.

3 Marco Baroni, Georgiana Dinu and German Kruszewski, Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors, in *ACL (1)*, 2014, pp. 238–247.

4 Yoshua Bengio, Rejean Ducharme, Pascal Vincent and Christian Jauvin, A neural probabilistic language model, *Journal of Machine Learning Research* 3 (2003), 1137–1155.

5 Bob Coecke, Mehrnoosh Sadrzadeh and Stephen Clark, Mathematical foundations for a compositional distributional model of meaning, arXiv:1003.4394, 2010.

6 George Cybenko, Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals, and Systems (MCSS)* 2(4) (1989), 303–314.

7 Jeffrey L. Elman, Finding structure in time, *Cognitive science* 14(2) (1990), 179–211.

8 John R. Firth, *A Synopsis of Linguistic Theory, 1930–1955*, Oxford University Press, 1957.

9 Gottlob Frege, Über sinn und Bedeutung, *Zeitschrift für Philosophie und philosophische Kritik*, 1892.

10 L. T. F. Gamut, *Logic, Language, and Meaning. Volume II. Intensional Logic and Logical Grammar*, University of Chicago Press, 1991.

11 Felix A. Gers and Jürgen Schmidhuber, LSTM recurrent networks learn simple context-free and context-sensitive languages, *IEEE Transactions on Neural Networks* 12(6) (2001) 1333–1340.

12 Christoph Goller and Andreas Küchler, Learning task dependent distributed representations by backpropagation through structure, in *IEEE International Conference on Neural Networks, 1996*, Vol. 1, IEEE, 1996, pp. 347–352.

13 Alex Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, PhD thesis, Technical University Munich, 2008.

14 Alex Graves, Generating sequences with recurrent neural networks, arXiv:1308.0850, 2013.

15 Irene Heim and Angelika Kratzer, *Semantics in Generative Grammar*, Vol. 13, Blackwell, 1998.

16 Geoffrey E. Hinton, Learning distributed representations of concepts, in *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Vol. 1, Amherst, MA, 1986, p. 12.

17 Sepp Hochreiter and Jürgen Schmidhuber, Long short-term memory, *Neural computation,* 9(8) (1997), 1735–1780.

18 Kurt Hornik, Maxwell Stinchcombe and Halbert White, Multilayer feedforward networks are universal approximators, *Neural Networks* 2(5) (1989), 359–366.

19 Phong Le and Willem Zuidema, Compositional distributional semantics with long short term memory, *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM)*, 2015.

20 Omer Levy and Yoav Goldberg, Neural word embedding as implicit matrix factorization, in *Advances in Neural Information Processing Systems*, 2014, pp. 2177–2185.

21 Tomas Mikolov, *Statistical Language Models Based on Neural Networks*, PhD thesis, Brno University of Technology, 2012.

22 Tomas Mikolov, Kai Chen, Greg Corrado and Jerffey Dean, Efficient estimation of word representations in vector space, arXiv:1301.3781, 2013.

23 Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado and Jeff Dean, Distributed representations of words and phrases and their compositionality, in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

24 Tomas Mikolov, Wen-tau Yih and Geoffrey Zweig, Linguistic regularities in continuous space word representations, in *Proceedings of NAACL-HLT*, Vol. 13, 2013, pp. 746–751.

25 Jeff Mitchell and Mirella Lapata, Vector-based models of semantic composition, in *Proceedings of the ACL*, 2008, pp. 236–244.

26 Jeff Mitchell and Mirella Lapata, Composition in distributional models of semantics, *Cognitive Science* 34(8) (2010), 1388–1429.

27 Saif M. Mohammad, Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney, Computing lexical contrast, *Computational Linguistics* 39(3) (2013), 555–590.

28 Richard Montague, English as a formal language, in Bruno Visentini et al., eds., *Linguaggi nella società e nella tecnica*, Edizioni di Communita, 1970, pp. 189–224.

29 Richard Montague, Universal grammar, *Theoria* 36(3) (1970), 373–398.

30 Richard Montague, The proper treatment of quantification in ordinary English, in *Approaches to Natural Language*, Springer, 1973, pp. 221–242.

31 Barbara H. Partee, Montague grammar, in Neil J. Smelser and Paul B. Baltes, eds., *International Encyclopedia of the Social & Behavioral Sciences*, Vol. 11, Elsevier, 2001.

32 Jordan B. Pollack. Recursive distributed representations, *Artificial Intelligence* 46(1) (1990), 77–105.

33 Michael Repplinger, *Understanding Generalization: Learning Quantifiers and Negation with Neural Tensor Networks*, Master Thesis, Master of Logic, University of Amsterdam, 2017.

34 Richard Socher, Christopher D. Manning and Andrew Y. Ng, Learning continuous phrase representations and syntactic parsing with recursive neural networks, in *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop,* 2010, pp. 1–9.

35 Kai Sheng Tai, Richard Socher and Christopher D Manning, Improved semantic representations from tree-structured long short-term memory networks, *Association for Computational Linguistics 2015 Conference*, 2015.

Sander M. Bohté
Machine Learning Group
Centrum Wiskunde & Informatica, Amsterdam
s.m.bohte@cwi.nl

Davide Zambrano
Machine Learning Group
Centrum Wiskunde & Informatica, Amsterdam
d.zambrano@cwi.nl

Adapting spiking neural networks

Understanding how neurons are able to efficiently encode information is a topic with applications ranging from more efficient neural network chips, to robot control, and also to future prosthetics that directly communicate with neurons. To understand how the brain is able to operate efficiently and asynchronously, Sander Bohté and Davide Zambrano describe models of biological neurons and examine how the spiking nature of neuronal communication relates this question.

A central goal of Artificial Intelligence (AI) is to develop algorithms that match the human ability to perceive, plan and act, be it vision, hearing, smelling, or touching. The human brain shows a remarkable ability to deal with the difficult task of making sense of the external world. This task is difficult in many ways: perception itself is by definition noisy and ambiguous, and muscles are notoriously hard to control as factors like fatigue, growth and atrophication alter the effect of motor commands given by the brain.

Loosely modeled after the neuronal networks of the brain, so-called deep neural networks have revolutionised AI in recent years, delivering breakthrough performance on such diverse tasks like image recognition, speech recognition, and superhuman performance playing Go and Chess: we have truly entered the age where computers are better at certain 'intelligent' tasks than humans. Here, we will give some intuition into the question of what these deep neural networks are, how they relate to the brain, and what more we can learn from the brain.

The human brain consists of an intricate web of billions of interconnected cells called 'neurons', where each neuron typically makes connections to up to $10{,}000$ other neurons. The study of neural networks in computer science aims to understand how such a large collection of connected elements can produce useful computations, such as vision and speech recognition, and also motor control, like using perception to catch a ball.

Artificial neural networks

Artificial neural networks (ANNs) are abstractions of the computation believed to be carried out by real neurons. A 'real' neuron receives pulses from many other neurons. These pulses are processed in a manner that may result in the generation of puls-

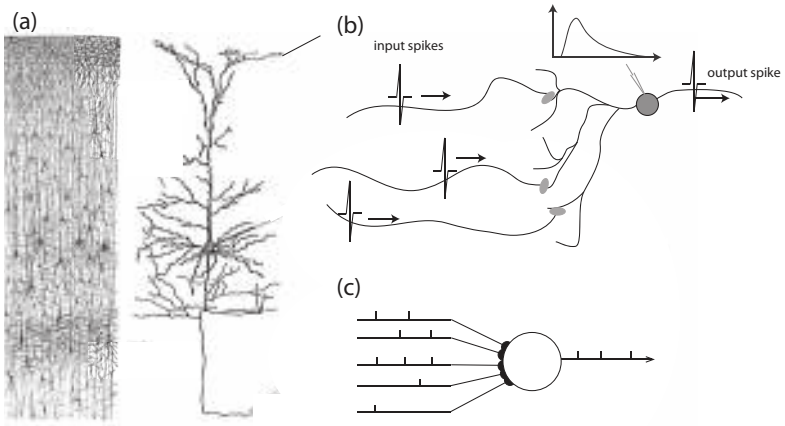


Figure 1 (a) Staining of just some of the neurons in a piece of cortex. A singly (pyramidal) neuron is shown as well. Notice the many synapses where the neuron receives input from other neurons. (b) Neurons communicate with each other using spikes, which each influence the internal state (typically the membrane potential) of the target neuron. (c) Abstracted representation of spike-based communication, where synapses between neurons are taken as 'weights'.

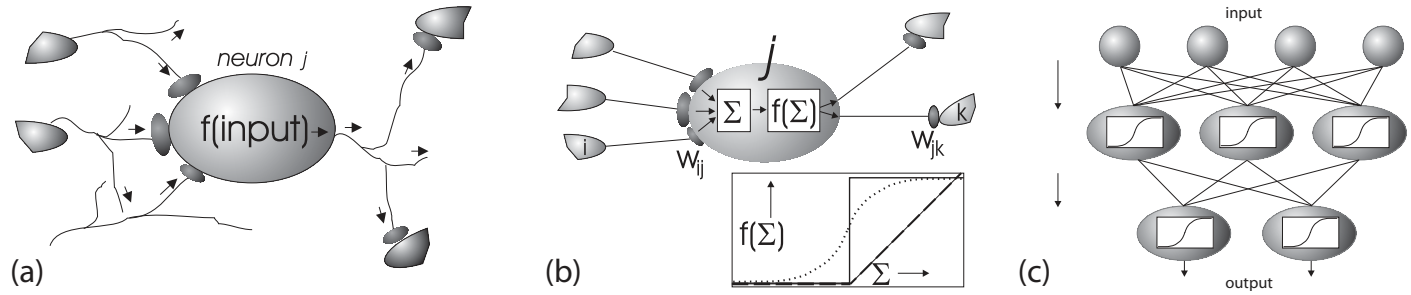


Figure 2 (a) A neuron computes output from inputs. (b) Artificial neuron modeling input–output computation. Inset: transfer functions $f(\Sigma)$. Plotted are examples of a binary (solid line), sigmoidal (dotted line) and ReLU (dashed line) transfer function. (c) Stringing neurons together to obtain a multilayer perceptron network.

es in the receiving neuron, which are then transmitted to other neurons (Figure 1b, c). The neuron thus 'computes' by transforming input pulses into output pulses.

ANNs try to capture the essence of this computation: as depicted in Figure 2, the rate at which a neuron fires pulses is abstracted to a scalar 'activity value', or *output*, assigned to the neuron. Directional connections determine which neurons are input to other neurons. Each connection has a weight, and the output of a particular neuron is a function of the sum of the weighted outputs of the neurons it receives input from. The applied function is called the *transfer function*, $F(\Sigma)$. Binary 'thresholding' neurons have as output a '1' or a '0', depending on whether or not the summed input exceeds some threshold. Sigmoidal neurons apply a sigmoidal transfer function, and have a real-valued output, and so-called 'rectified linear' neurons, or 'ReLU') neurons apply a rectified linear function (inset Figure 2b, solid respectively dotted and dashed line). Abstracted in the sigmoidal transformation function is the idea that real neurons communicate via firing rates: the rate at which a neuron generates action potentials (spikes). When receiving an increasing number of (positively weighted) spikes, a neuron is naturally more likely to emit an increasing number of spikes itself.

Neural networks are sets of connected artificial neurons. Remarkably, networks of such simple, connected computational elements can implement a range of mathematical functions relating input states to output states, where its computational power is derived from the connectivity pattern and clever choices for the values of the connection weights.

Learning rules for neural networks prescribe how to adapt the weights to improve performance given some task. An example of a neural network is the *multi-layer perceptron* (MLP, Figure 2c). Learning rules like *error backpropagation* [23] compute the gradient of each weight with respect to a pre-defined loss function that captures the cost of deviations from desired behavior. The weights in the network are adjusted along this gradient to minimize the loss, which enables the neural networks to learn and perform many tasks associated with intelligent behavior, like learning, memory, pattern recognition, and classification [1, 22].

Different types of neural networks have been developed over the last two decades. So-called convolutional neural networks take their inspiration from filters used in computer vision: a filter is specified as a matrix of weights which is convolved with an image to, for instance, detect edges in figures. The benefit of this approach is that a single small filter, like a 3×3, 4×4 or 5×5 matrix of weights, can be applied to the entire image: detecting edges thus needs only few parameters (the weights). Applying the same filter to the entire image also makes sense, as edges can be anywhere in the image.

Convolutional neural networks (CNNs) exploit this principle, having small filters that are convolved with the image, however, rather than handcrafted, the filters are learned. The convolutional neural networks, as illustrated in Figure 3a, moreover contains many filters in a layer, and a layer of filters connects to a next layer of filters, to learn filters-of-filters. In deep CNNs, many of these layers are used successively. The resultant filters-of-filters become sensitive to progressively complex features in the image, like from edges to lines, to features like noses, mouths and eyes, to faces. It was this type of neural network, developed already in the late 1980's by Yann LeCun, that achieved the breakthrough in

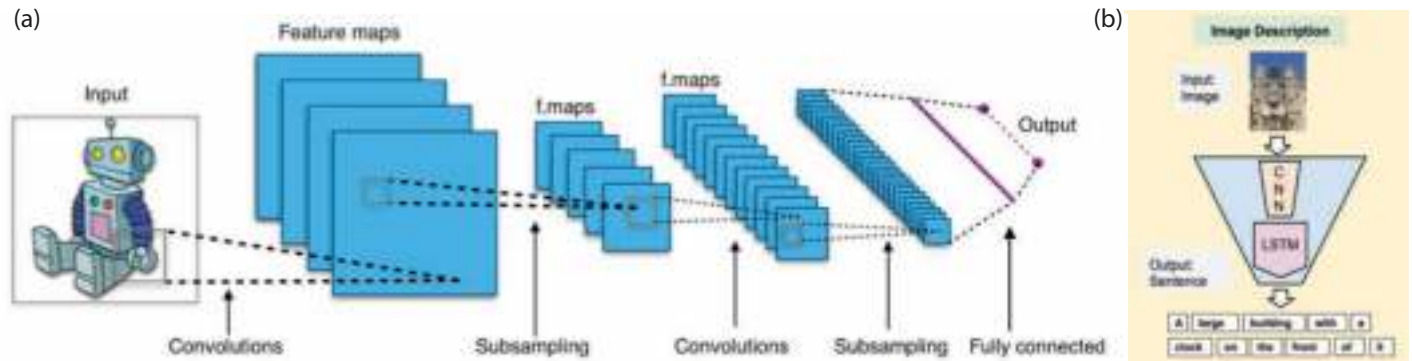


Figure 3 (a) Deep convolutional neural networks. Feature layers extract features from previous maps, while subsampling layer compress features maps to create more position-invariant features. (b) A neural network that writes captions to describe an image: an image is parsed by a CNN and the output of the CNN is parsed by an LSTM to select a sequence of words (taken from [8]).

object classification by Alex Krizhevsky and Geoffrey Hinton in 2012.

Convolutional neural networks are complemented with neural network structures capable of learning to maintain information-memory structures. Many tasks have a sequential nature where information has to be integrated and maintained to make the right inferences or choices: from reading a text to driving from home to work. While recurrent network structures can in principle (learn to) maintain relevant information, it was found in the late 1980's that such structures are notoriously hard to train. In 1998 then, Sepp Hochreiter and Jürgen Schmidhuber developed a memory structure with more tractable learning properties, so-called Long Short-Term Memory, or LSTM. Such networks, and variants thereof, are the workhorse of modern neural networks for sequential tasks. Figure 3b shows an impressive example of how convolutional neural networks and LSTMs are combined to create remarkably accurate captions for images.

Much of the magic of modern neural networks is enabled by the development of very powerful hardware for computing the matrix multiplications that underly the computations in neural networks. The star here are GPUs: initially developed for high-end gaming, it turned out that the massively parallel hardware was an excellent fit for computing neural networks. Only the last few years has dedicated AI hardware started to emerge, ranging from ultra-high performance tensor processing units (TPUs) developed by Google, to dedicated 'AI' blocks in cell-phone chips like Huawei's Kirin 970. Exploiting this powerful hardware has also become feasible by the development of high-level neural network frameworks, like Tensorflow and PyTorch, that make it easy to implement and train neural networks in an almost hardware agnostic manner.

Back to the brain

While deep neural networks are achieving huge successes, in many aspects they still pale in comparison to their biological source of inspiration, the brain. For example, the brain needs vastly fewer examples to learn tasks and is massively more energy efficient, while it's ability to control hundreds of flexible and variable muscles for motion remains unsurpassed. In particular, artificial neural networks operate according to a synchronised paradigm of computation, where in a single pass all neurons exchange their activation values and update their internal state. In contrast, the brain operates in an asynchronous fashion: real neurons only exchange information when they receive sufficient inputs, and they do so only rarely. Understanding how neurons are able to efficiently encode information is a topic with applications ranging from more efficient neural network chips, to robot control, and also to future prosthetics that directly communicate with neurons — neuroprosthetics. Thus, to understand how the brain is able to operate efficiently and asynchronously, we return to our models of biological neurons, and in particular examine how the spiking nature of neuronal communication relates this question.

The question of how neurons encode information in the spikes they emit is a hotly debated one in neuroscience. At the heart of the argument is the issue to what degree neurons respond in a stochastic manner to received inputs: on the one hand, many experimental findings show that neuronal firing is highly unreliable, and can be reasonably described as a rate-driven Poisson process. On the other hand, we know that individual spiking neurons can be highly reliable, emitting reproducible spikes at a very high time resolution [15]. Part of the reason for this finding seems to be that spike-time reliability is related to the temporal properties of the received signal; an example of this phenomenon is shown in Figure 4.

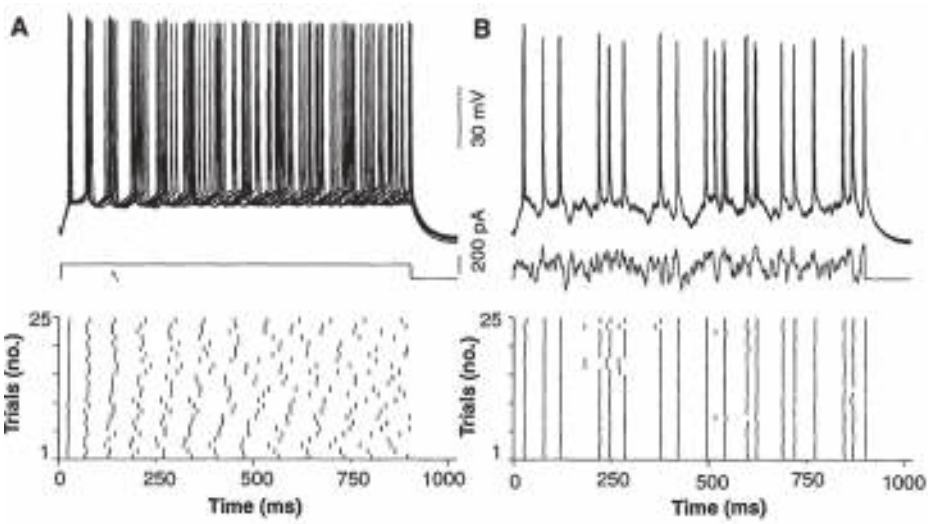


Figure 4 Reliability of firing of real spiking neurons when inject by either a constant (a) or fluctuating current (b). Top: overlapped voltage traces from 25 trials obtained from a single neuron repeatedly injected with a either a fixed or variable current profile (middle). Bottom: raster-plot of individual spike-times for each of the 25 trials. Note the dispersion in spike-times for a fixed current injection and the reliability of spike-timing for the fluctuating current profile. Graph taken from [15].

Spike times carrying significant information is attractive, as in theory it increases the amount of information carried by each spike. Information theoretic measurements on the entropy of in-vivo (real-world) neurons have also shown significant information in the precise spike timing [16].

Spiking neuron models

The prototypical model of a spiking neuron is the Hodgkin–Huxley model. Experimenting on the squid's giant axon, Hodgkin and Huxley [13] found that three ionic currents determined most of the axon's behavior: the sodium and potassium currents, and a leak current. Ion channels in the neuron's cell membrane control the flow of ions in a voltage-dependent manner, where the interior of the cell acts as a capacitor. This leads them to propose a relatively simple electrical circuit as a model of the neurons response to a current entering the cell (Figure 5a), where the current partly charges the capacitor and partly leaks through the ion-channels.

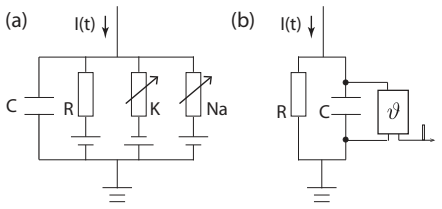


Figure 5 (a) Electrical circuit for the Hodgkin–Huxley neuron model. (b) Electrical circuit for the leaky integrate-and-fire neuron model.

In this model, mathematically, we split an applied current $I(t)$ into a current charging the capacitor I_{cap} and components I_k leaking trough the ion channels:

$$I(t) = I_{\text{cap}} + \sum_k I_k(t)$$

For a voltage u across the capacitor, we can substitute $I_{\text{cap}} = C du/dt$, rewriting:

$$C\frac{du}{dt} = -\sum_k I_k(t) + I(t).$$

For the leakage currents $I_k(t)$, Hodgkin and Huxley formulated differential equations for the three main components, the voltage dependent Na^+, K^+ channels and a generic leakage channel:

$$\sum_k I_k = g_{\text{Na}} m^3 h(u - E_{\text{Na}}) + g_{\text{K}} n^4 (u - E_{\text{K}}) + g_{\text{L}}(u - E_{\text{L}}),$$

where $E_{\text{Na}}, E_{\text{K}}$ and E_{L} are the respective reversal potentials; $g_{\text{Na}}, g_{\text{K}}$ and g_{L} are the respective maximum channel conductances. The variables m, h and n are the gating variables that control the Na^+ and K^+ channels, and that evolve as:

$$\frac{dm}{dt} = \alpha_m(u)(1-m) - \beta_m(u)m, \quad (1)$$

$$\frac{dn}{dt} = \alpha_n(u)(1-n) - \beta_n(u)m, \quad (2)$$

$$\frac{dh}{dt} = \alpha_h(u)(1-h) - \beta_h(u)m. \quad (3)$$

Hodgkin and Huxley then fitted the functions $\alpha(u)$ and $\beta(u)$ to the experimental data.

The Hodgkin and Huxley equations provide an accurate description of many dynamical responses of the squid axon, and by choosing different values for the various variables, many types of observed neural responses can be fitted. For example, a current injection may result in a moderate disturbance of the membrane potential, the generation of a single spike, or even trigger a burst of spikes persisting for much longer than the current injection.

Still, while the equations can be studied with the tools of mathematics, the behavior of such high-dimensional and non-linear equations is both hard to analyze and hard to visualize.

Leaky integrate-and-fire (LIF)

To study topics like memory and neural coding, simple phenomenological models are often preferred. Such models capture both the dynamics of the membrane potential as a function of impinging spikes and current injections while also prescribing the conditions for a neuron to generate an action potential.

Broadly, the transmission of a single spike from one neuron to another is mediated by *synapses* at the point where the two neurons interact. An input, or *presynaptic* spike arrives at the synapse, which in turn releases neurotransmitter which then influences the state, or *membrane potential* of the target, or *postsynaptic* neuron. When the value of this state crosses some threshold ϑ, the target neuron generates a spike, and the state is reset by a *refractory response*. The size of the impact of a presynaptic spike is determined by the type and efficacy (weight) of the synapse (this is illustrated in Figure 6).

The electrical circuit describing a Leaky-Integrate-and-Fire (LIF) neuron is a simple version of the Hodgkin-Huxley neuron: as illustrated in Figure 5b, it consists of a current $I(t)$ driving a capacitor C in parallel with a resistor R. The current again splits in a component that charges the capacitor and one that passes through the resistor: $I(t) = I_{\text{cap}} + I_R$. Substituting $I_R = u/R$ (Ohm's Law) and $C = q/u$, where u is the voltage and q is the charge, we get:

$$I(t) = \frac{u(t)}{R} + C\frac{du}{dt}.$$

With $\tau_m = RC$, we can rewrite this as:

$$\tau_m \frac{du}{dt} = -u(t) + RI(t),$$

where we identify $u(t)$ as the membrane potential of the neuron, and τ_m as the membrane time constant.

The neuron emits a spike when the membrane potential reaches a threshold ϑ from below. When this happens, the membrane potential is reset to a new value $u_r < \vartheta$. This combination of leaky integration of incoming current and a reset at the time of spiking characterises LIF neuron models.

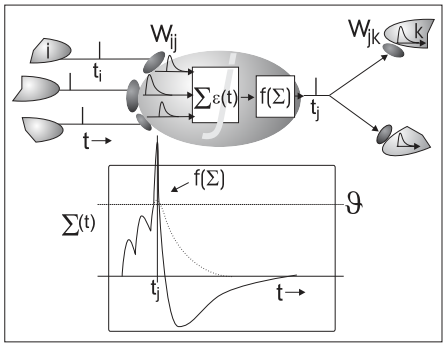


Figure 6 Impact of spikes on the potential of a target neuron.

Many different variations of LIF neurons can be created, including versions that include refractory effects at the time of spiking, where the threshold is stochastic and where the threshold is dynamic. Such more elaborate models of LIF neurons have been shown to predict neural behavior to a remarkable degree when compared to experimental data; a wonderful and accessible treatise on this topic has been written by Gerstner and Kistler [10].

Synaptic currents

So far, spiking neurons have been described in terms of their response to currents $I(t)$ injected into the neuron. In reality, these currents are caused by neurotransmitters arriving through synapses triggered by spikes from the sending neuron (presynaptic spikes).

Synapses are complicated beasts: broadly, neurotransmitters are released in quantiles, contained in small vesicles that release their content by fusing with the synapse' membrane. This process seems to be both stochastic and history-dependent: the amount of neurotransmitter released at a synapse in response to the arrival of a spike can vary dramatically.

Ignoring this complexity, we can model the current that a presynaptic spike at time t_j contributes to a postsynaptic neuron i as a post-synaptic current (PSC) with time course $\alpha(t - t_j)$ weighted by a particular weight, or 'synaptic efficacy w_{ij}. A neuron i thus receives as input currents:

$$I(t) = \sum_j w_{ij} \sum_{t_j} \alpha(t - t_j).$$

The most simple model for the postsynaptic current $\alpha(s)$ is a Dirac $\delta-$pulse, $\alpha(s) = q\delta(s)$, for a total current contribution q. More realistic models let the current α have a finite duration, for example an exponential decay with time constant τ_s:

$$\alpha(s) = \frac{q}{\tau_s}\exp\left(-\frac{s}{\tau_s}\right)\Theta(s),$$

where $\Theta(s)$ denotes the Heaviside step function.

Spike Response Model (SRM)

Wulfram Gerstner [11] developed the Spike

Response Model (SRM) as a non-linear integrate-and-fire model that expresses the membrane potential at time t as an integral over the past, as opposed to a formulation in terms of dynamical systems. Specifically, the membrane potential is modelled as a sum of (weighted) impinging post-synaptic potentials, $\epsilon(t)$, and refractory responses $\eta(t)$:

$$u_i(t) = \sum_{t_i} \eta(t - t_i) + \sum_j w_{ij} \sum_{t_j} \epsilon(t - t_i, t - t_j),$$

where ϵ and η are *response kernels*. The threshold, that determines when a neuron fires, can be dynamical: $\vartheta \longrightarrow \vartheta(t - t_i)$. Many phenomenological models include such dynamical thresholds to explain the spiking behavior of many different neurons. The main benefit of the SRM formulation is that in many ways, SRM formulations of LIF-neurons are much more easily interpretable. We will rely on this in our formulation of a spike-time-based neural code later on.

Neural networks

As an artificial neuron models the relationship between the inputs and the output of a neuron, artificial spiking neurons describe the input in terms of single spikes, and how such input leads to the generation of output spikes. For this, we need to relate spikes to information and computation.

As noted, the exact nature of neural coding by biological neurons is still unresolved in neuroscience and subject to much debate. A recent line of work suggests that spiking neurons may implement adaptive online analog-to-digital and digital-to-analog (AD/DA) conversion [2, 3, 4, 6, 26]: the key observation is that when a neuron spikes, the refractory reset removes a part of the internally computed analog voltage signal, which the spike, through the synapse, delivers to the next neuron. Young [26] recently demonstrated a direct correspondence between simple leaky integrate-and-fire (LIF) models and the AD/DA encoding/decoding scheme in electrical engineering called *asynchronous pulse sigma-delta modulation* (APSDM). The APSDM scheme however presumes a fixed dynamic range for the encoded analog values, as signals are 'chopped' into fixed-size pieces, and requires that a neuron fires at a very high firing rate to obtain a good signal approximation.

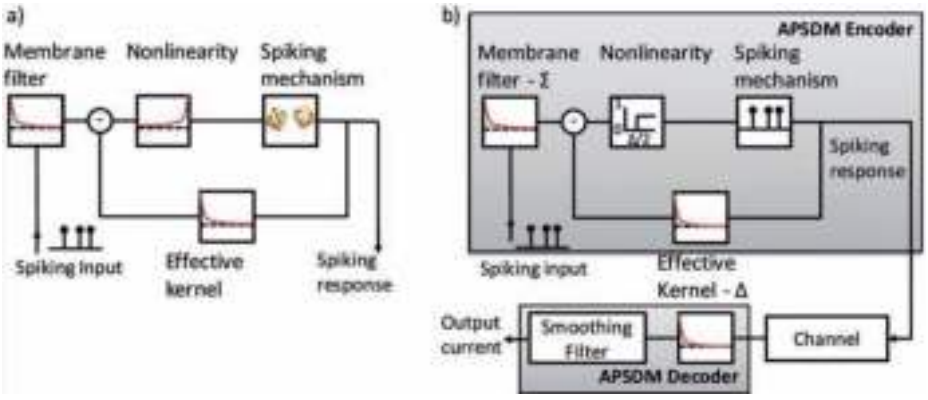


Figure 7 (a) Generalized leaky integrate-and-fire and (b) the asynchronous pulse sigma-delta modulation (APSDM) [26]. The APSDM scheme consists of an encoder (analog-to-digital), a channel, and a decoder (digital-to-analog), where signals are encoding using uni-polar pulses (spikes). A signal is first passed through a filter Σ and added to the internal state. Then a non-linearity is applied in the form of a thresholding function with threshold $\Delta/2$. When the threshold is exceeded, a pulse is sent to the decoder through the channel, while a response kernel Δ is subtracted from the internal state of the neuron. At the decoder, each pulse is decoded with a fixed response kernel and then smoothed. Note the close similarities between the APSDM scheme and the LIF neuron on the left.

Adaptive spike coding

We can combine the APSDM scheme with a spiking neuron model that dynamically adapts to the (varying) dynamic range of the computed internal activation value. Inspired by [5] and [3], we use a multiplicative model of adaptation to obtain a spiking neuron that is capable of encoding and decoding a wide dynamic range of activation values with a limited firing rate. We show that we can thus compose computationally efficient adaptive spiking neural networks through drop-in replacement of analog neurons in artificial neural networks (ANNs), which achieve identical performance to these ANNs without additional modifications.

A spiking neural network is defined by the relationship between spikes and the quantity that is computed in the neuron as the result of impinging spikes. With *adaptive spike-time coding*, weighted input spikes contribute linearly to the membrane potential, and when this sum of inputs reaches (positive) threshold, a spike is generated while a refractory reset is subtracted from the membrane potential. Since both refractory reset and the contribution of impinging spikes are temporally extended, intuitively, the (smoothed) sum of refractory resets corresponds to the signal conveyed to the next neuron; this process is illustrated in Figure 8a.

Adaptive spiking neuron using multiplicative adaptive spike-time coding

To create artificial spiking neural networks based on adaptive spike-time coding, we address the limited dynamic range of standard LIF or corresponding Spike Response Model (SRM) neurons. We note that it is the fixed size refractory resets that limit the dy-

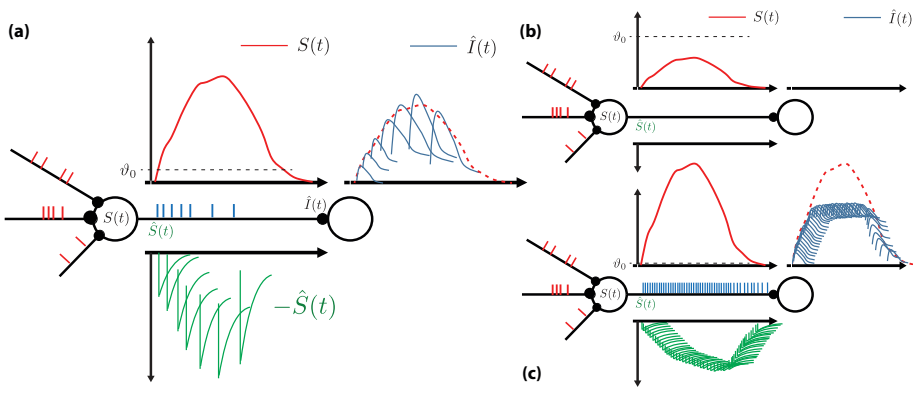


Figure 8 (a) Illustration of signal encoding with the ASN. $\hat{I}$ denotes the smoothed sum of (weighted) postsynaptic currents in the post-synaptic target neuron, proportionally approximating the encoded presynaptic signal $S(t)$. (b,c) Limited dynamic range: approximations fail when the signal $S(t)$ is too small relative to the neurons' threshold ϑ_0 (no spikes), or, (c) too large: then, due to absolute refractoriness and corresponding maximum firing rate, the 'high' parts of the signal $S(t)$ cannot be encoded.

namic range of the internal activation that a neuron can encode [3, 6]. Effectively, activation values that are either too small or too large relative to the threshold cannot be encoded. We use the solution proposed in [3] based on fast adaptation: by dynamically adjusting the threshold, the size of the refractory responses can be controlled and the dynamic range can be increased, drastically even when a multiplicative form of threshold adjustment is used. Such multiplicative adaptation effectively allows a neuron to assign a fixed 'budget' of spikes to a given dynamic range, also when that range changes drastically. Such a model of adaptation also explains various adaptive behaviour in real biological neurons [3, 5, 9].

We implement adaptive spike-time coding using multiplicative adaptation in an SRM [10]. A spiking neuron computes a smoothed internal activation value $S(t)$ on the input current:

$$S(t) = (\phi * I)(t),$$

where $\phi(t)$ is the (exponential) smoothing filter with time constant τ_{smooth} and $I(t)$ is the input current that the neuron receives. This current $I(t)$ can be injected directly into the spiking neuron (for inputs), or be the result of impinging (weighted) spikes causing post-synaptic currents (PSCs) (specified below). The spiking mechanism approximates the ReLU activation of $S(t)$ with $\hat{S}(t)$ using a sum of spike-triggered kernels $\eta(t-t_i)$:

$$\hat{S}(t) = \sum_{t_i} \eta(t-t_i), \tag{4}$$

where a spike is added in an online and incremental fashion when the difference between the input signal and the signal approximation exceeds a positive dynamic threshold $\vartheta(t)$ from below:

$$u(t) = S(t) - \hat{S}(t) > \vartheta(t), \tag{5}$$

where $u(t)$ denotes the neuron's membrane potential. Upon emitting a spike at t_i, the spike-triggered refractory response $\eta(t-t_i)$ is subtracted from $S(t)$ and added to $\hat{S}(t)$. The part of $S(t)$ larger than the minimal value of the threshold $\vartheta(t)$ is thus encoded as $\hat{S}(t)$ in a spike train t_i. It is decoded at the postsynaptic target neuron where the resultant postsynaptic currents are added as weighed versions of the refractory response $\eta(t)$. The resultant postsynaptic current in target neuron j, $I_j(t)$ induced by presynaptic spikes t_i from multiple presynaptic neurons i, is then computed as:

$$I_j(t) = \sum_i \sum_{t_i} w_{ij} \eta(t-t_i),$$

where w_{ij} is the weight between presynaptic neuron i and postsynaptic neuron j. The refractory response kernel $\eta(t)$ is adaptive and controlled through the dynamic threshold $\vartheta(t)$:

$$\eta(t-t_i) = \vartheta(t_i)\, \varkappa(t-t_i),$$

where $\vartheta(t_i)$ is the effective threshold at the time of spiking, $\varkappa(t-t_i)$ is a spike-triggered exponentially decaying response kernel shaping the refractory response due to the spike at t_i with normalised height $\varkappa(0) = 1$. Thus computed, the average of the sum of η kernels approximates the mean of the (rectified positive) signal $S(t)$.

We model the dynamic threshold $\vartheta(t)$ as multiplicative adaptation after [3]:

$$\vartheta(t) = \vartheta_0 + \sum_{t_i} m_f \vartheta(t_i)\, \gamma(t-t_i), \tag{6}$$

where ϑ_0 is the baseline threshold set to some (small) fixed value. A multiplicative factor m_f of fixed size regulates the threshold dynamics, where the ratio between ϑ_0 and m_f determines the asymptotic firing rate of the neuron for large activation values. The adaptation kernel $\gamma(t)$ is computed as a sum of exponentials: $\gamma(t) = \sum_n \gamma_n \exp(-t/\tau_{\gamma_n})$ with weights γ_n normalised to one such that $\gamma(0) = 1$. A few components are sufficient to mimic limited long-memory adaption as experimentally reported in e.g. [21]; here we use either one or two components. Note that the internal state of the neuron is fully determined by the two kernels $\eta(t)$ and $\gamma(t)$, and both of these kernels can be expressed as sum of exponentials: one for $\varkappa(t)$ and one or two for $\gamma(t)$. The neuron state update can thus be efficiently computed by updating these exponential functions as simple (memory-less) dynamical systems.

As noted, the signal approximation $\hat{S}(t)$ is computed as a sum of variable height kernels: it is this signal that is communicated through a sequence of spikes to the next, postsynaptic, neuron. At the postsynaptic neuron, a filter $\phi(t)$ smooths the (weighted) η kernels, which suppresses high frequency noise and reconstructs the signal as in the APSDM receiver [26]. In the network, for each arriving spike the corresponding η kernel is multiplied by the weight of the connection and added to the current $I(t)$ in the post-synaptic neuron. Since the height of the η kernel is adaptive, in this treatment each spike t_i effectively has a *height* $\vartheta(t_i)$. Thus, the ASN communicates spikes with an analog 'height' rather than binary valued spikes.

Adaptive signal encoding and decoding

The (unsmoothed) signal approximation $\hat{S}(t)$ computed by the spiking mechanism in the adaptive spiking neuron computes a ReLU function: plotted in Figure 9a is both the firing rate (dashed) and the mean and standard deviation of the signal approximation $\hat{S}(t)$ (solid) for increasing signal values S, for two different ratios of ϑ_0 and m_f. While the firing rate saturates, the approximation $\hat{S}(t)$ remains linearly growing with increasing S, albeit with increasing variance as the number of spikes used to encode the signal remains the same.

Since the ratio of the baseline threshold ϑ_0 and the multiplicative factor m_f determines the saturating firing rate, this ratio also determines the precision of the en-

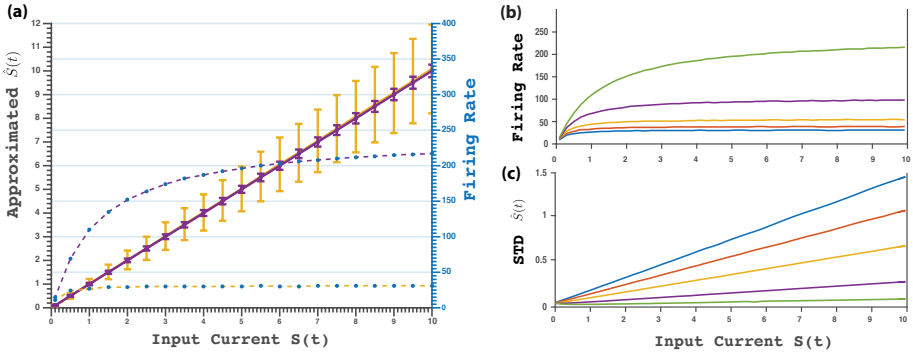


Figure 9 (a) Firing rates (dashed lines, right axis, computed over a 1 s time window), output signal $\hat{S}(t)$ with standard deviation (solid lines, left axis) of an ASN ReLU neuron for two firing rate regimes ($\vartheta_0 = 0.1$, $m_f = \vartheta_0$ (yellow), $m_f = 0.1\vartheta_0$ (purple)). Colors are the same for firing rate and corresponding signal $\hat{S}$. (b) Firing rate for 5 different values of $m_f = 0.01, 0.025, 0.05, 0.075, 0.1$ and $\vartheta_0 = 0.1$. (c) Standard deviation (std) for 5 different values of m_f. Colors correspond between (b) and (c).

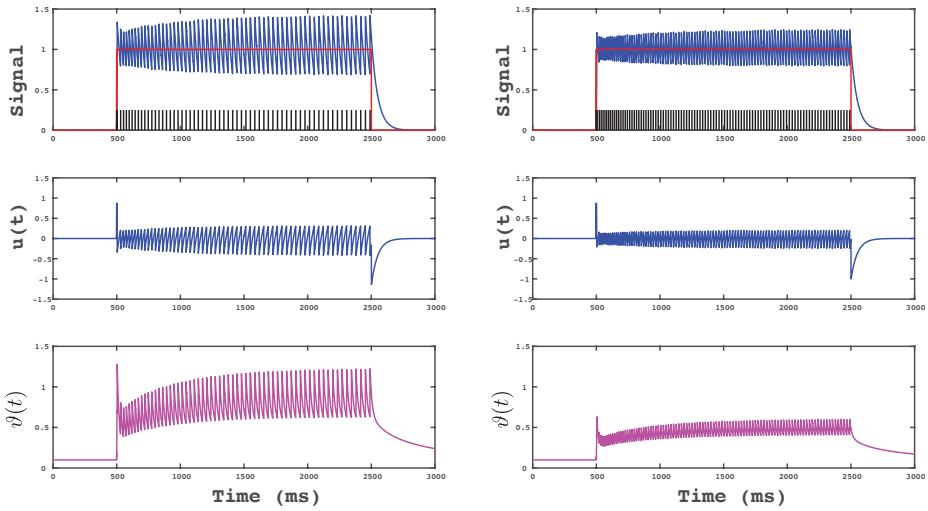


Figure 10 Encoding of two fixed size step functions for $S(t)$, illustrating the decreasing variance of the signal approximation $\hat{S}(t)$ for increasing firing rates. Parameters: $m_f = 1\vartheta_0$ (left) and $0.1\vartheta_0$ (right) for $\vartheta_0 = 0.1$.

coding. The inverse relationship between saturating firing rate and coding precision is plotted in Figure 9b,c for five different values of m_f/ϑ_0. We observe that the standard deviation linearly increases with signal magnitude, and inversely relates to the saturating firing rate.

In Figure 10, we illustrate signal encoding with the ASN with more or less spikes. In the top row we plot the encoding of a step function $S(t)$ (red) with a sum of adaptive kernels, $\hat{S}(t)$ (blue). The black dashes denote the spikes: the variance of $\hat{S}(t)$ decreases when more spikes are used. In the middle row, the membrane potential $u(t)$ is plotted for both cases, and in the bottom row the dynamical threshold $\vartheta(t)$. As can be seen, a lower firing rate is achieved by a higher average threshold and correspondingly larger refractory resets $\eta(t)$.

The time constant of the refractory response $\eta(t)$ is determined by τ_χ: the value of this constant determines how much 'future' signal each spike transmits. To encode step functions as in Figure 10, a decay constant that better matches the temporal correlation in the approximated signal will yield a better approximation. For a step function, this effect is plotted in Figure 11. Shown is the sum squared error (SSE) approximating a 1 second segment of a step function with a fixed firing rate (35 Hz) for various values of τ_χ. Increasing τ_χ strongly reduces the SSE (blue line, left axis). The lower SSE however comes at the expense of responsiveness: when the step function steps back to 0, it takes longer before the approximation correctly matches the new, lower value. Plotted also (orange line, right axis) is the time it takes before the signal approximation is below 0.05 after stepping down.

Implementation

In the examples and in our network implementations, we use time constants that are roughly of the order of the corresponding values in biological spiking neurons, such as time constants of PSCs, membrane time constant and refractory response kernels, to obtain firing rates for active neurons in the range of 1–100 Hz, compatible with what is observed in biology. We use a time constant of $\tau_\chi = 50$ ms for the exponential decay of the χ kernel. The γ kernel was approximated as either a single decaying exponential or the sum of two exponentially decaying functions,

$$\gamma^1(t) = e(-t/\tau_{\gamma_1}),$$

or

$$\gamma^2(t) = \frac{1}{\gamma_1+\gamma_2}\left[\gamma_1 e(-t/\tau_{\gamma_1}) + \gamma_2 e(-t/\tau_{\gamma_2})\right],$$

with time constants $\tau_{\gamma_1} = 15$ ms and $\tau_{\gamma_2} = 100$ ms and respective weights $\gamma_1 = 0.1$ and $\gamma_2 = 0.01$. Adding additional components increases the long-memory adaptation behaviour of the ASN, but two components suffice here as we are not considering time-varying signals. We use a time constant of $\tau_{\text{smooth}} = 2.5$ ms for the signal reconstructing exponential smoothing filter $\phi(t)$ in all ASN units except for the output neurons. In the output units activity was filtered with an exponential filter with a longer time constant of $\tau_{\text{rout}} = 50$ ms, to compare activations between outputs for classification purposes. The simulations are computed with time steps of size 1 ms.

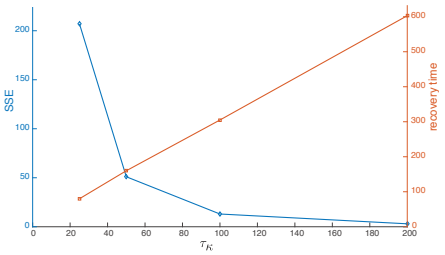


Figure 11 Error and responsiveness when encoding a step function with different $\eta(t)$ (or EPSP) time constants τ_χ. Left axis: sum-squared error. Right axis: responsiveness when switching back.

Adaptive spiking neural networks (ASNN)

We implement adaptive spiking neural networks where the units are comprised of the ASNs described above. Inherently, the ASNNs compute over time-continuous input signals; most straightforward and standard applications of deep neural networks are concerned with classification tasks, such as determining the digit in an image (Figure 12a). To compare classification performance between a standard ANN and an SNN, an image is presented for 500 ms to the network, and we record from the output neurons to determine the classification. The image is thus taken as input to the network for every time step in the SNN, which may be as small as 1 ms (1000 Hz) (illustrated in the inset in Figure 12b).

Since our ASNs communicate analog valued spikes rather than binary spikes, the question is how the classification problem thus phrased compares to a standard ANN which also communicates with analog values. For an image, an ANN can obviously compute the classification in one go, essentially using just one 'analog spike'. We argue that the correct comparison between SNNs, ASNNs and ANNs is to treat the classification problem as a time-continuous problem. While the stimulus is present the network has to compute classifications. For both SNNs and ASNNs this is inherent to the operation of the network, while an ANN would need to sample the input at a certain frame rate. This is illustrated in Figure 12b: the ANN computes the classification for each frame for the entire network, and the computational complexity scales linearly with the frame rate (illustrated in the right part of Figure 12b). In contrast, the SNN and ASNN implement an asynchronous model of ongoing neural computa-

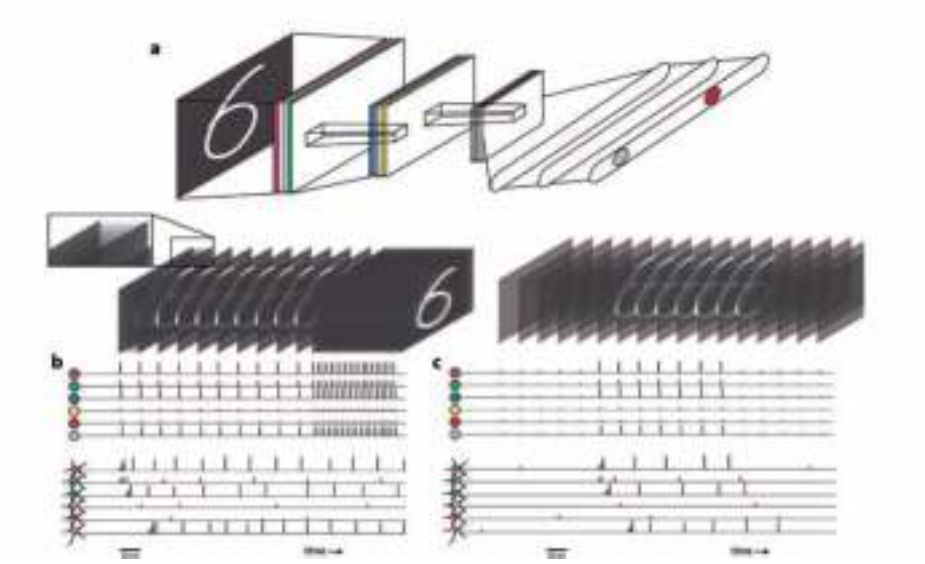


Figure 12 (a) deep convolutional neural network. (b) ANN versus ASNN classification. The ANN is computed for every frame, for the ASNN the neuron are updated at a fine resolution (inset), but network activity is asynchronous and sparse. Right part of the sequence: increasing the frame-rate increases ANN computations and not ASNN. (c) Flanked noise classification. The ANN computes at a fixed frame rate, also for noise input that activates feature neurons only slightly. For the ASNN, the input neurons rarely cross threshold and the network firing rate is very low for noise; spikes are only emitted when frames with features are presented.

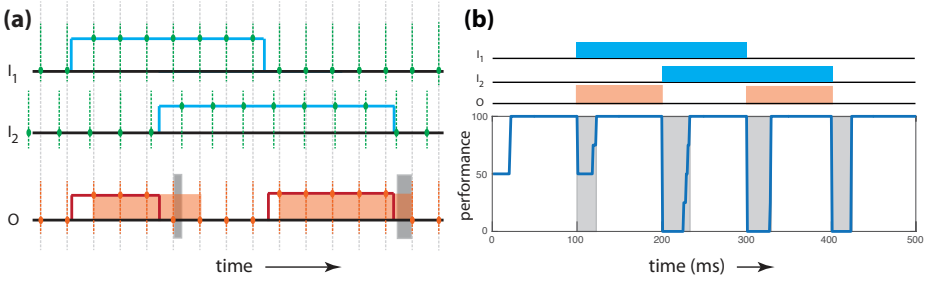


Figure 13 Asynchronously computing XOR: (a) illustration with inputs arriving asynchronously (dotted green lines), and XOR computed synchronously with the top (fastest) input rate. Due to the synchronous nature of computing, additional errors are made, like the shaded areas in the bottom figure. Processing the input asynchronously at their respective sample rates, the right shaded area would be avoided. (b) Asynchronous processing of XOR in a 2-5-1 ASNN network capable of computing XOR with about 15 Hz average firing rate and neurons using $\tau_x = 25$ ms. Novel input is processed at the update rate of the neurons (1 ms); the delay in classification when patterns switch is now determined by τ_x (shaded areas).

tion where the neurons are updated each small time step (1 ms), and communication between neurons is both localised (to active neurons), and a function of desired neural coding precision rather than frame-rate. Another benefit of the ASNN implementation is illustrated in Figure 12c: when no features are present in the frame, the spiking neural network does not generate spikes, or only very sparingly, whereas the ANN still computes the entire network every frame. The downside of asynchronous neural computation is that there is an inherent latency between input presentation and output: in each layer, the ASN applies an averaging filter to the spike-triggered input currents it receives.

Asynchronous neural computation offers benefits both for computing and for processing sensory motor data: with neural updating and network updating decoupled, sensory inputs (and actuator outputs) can be sampled at the high neural update frequency. This avoids the well known problem of synchronized processing [18]; the ASNN however cannot respond much faster to changing inputs than the τ_x time constant. This is illustrated in Figure 13 for the simple problem of streaming XOR: the network, using about a 15 Hz average firing rate, computes XOR from the two inputs. The bottom panel shows performance, and demonstrates that the network is still capable of responding faster to changes in input (≈ 25 ms) than a correspondingly synchronous sample rate.

Computational complexity

An examination of the computational cost and bandwidth requirements demonstrates the mixed ANN and SNN properties of the ASNN. In Table 1, these costs are specified. The ASNN shares the firing rate dependent network bandwidth cost with the SNN, but at an ANN-like cost per spike, and network delay is determined by the spike-decay time constant τ_x, (presumably) the same as in the SNN (not demonstrated in the literature). Since spike impact is computed as the product of spike height and connection weight, the ASNN shares the ANN's cost in terms of multiplications per spike/update, and the neuron update cost of the ASNN scales as an SNN.

This analysis ignores the fact that spikes in the ASNN (and SNN) are heavily localized to a subset of neurons: many neurons are silent while a few are active. Sparse and localised communication potentially offers a benefit to deep neural networks, as densely connected neural networks tend to be limited by the bandwidth required to read and write the appropriate weights from memory [24]. Thus reasoned, for an ASNN that incurs a 100 ms delay to compete in terms of bandwidth used with an ANN, it can use at most a firing rate of 10 Hz on average per neuron, since an ANN sampled with 10 Hz would achieve the same worst case delay. This ignores the benefit of the ASNN being able to process in principle a 1000 Hz frame rate. The exact benefit of sparse activity depends on the degree of sparseness and the degree to which parallel hardware can exploit sparseness.

	ANN	SNN	ASNN
Network bandwidth	$C \cdot [P+O] \cdot H_a$	$C \cdot O \cdot F_s$	$C \cdot [P+O] \cdot F_p$
Network delay	$1/H_a$	$\propto \tau_x + c \cdot L$	$\propto \tau_x + c \cdot L$
Network multiplications	$C \cdot P \cdot H_a$	—	$C \cdot P \cdot F_p$
Neuron multiplications	$H_a \cdot f(\mathrm{Re}LU)$	$U_s \cdot f(\mathrm{Re}LU)$	$U_p[3 + f(\text{threshold})]$

Table 1 Computational Cost. C: number of connections, P: pulse precision, H_a: ANN update frequency, O: addressing overhead, F_s: SNN firing rate, F_p: ASNN average firing rate, L: network depth (layers), U_s: update frequency of SNN, U_p: update frequency of ASNN, c: a constant.

Experimental networks

We demonstrate the ASNNs described above in fully connected feed-forward neural networks (FFNNs) and in a convolutional neural network (CNN) [14]. These architectures were first trained on standard datasets — IRIS, SONAR, and MNIST — with standard ANNs comprised of rectified linear (ReLu) neurons. The corresponding spiking neural networks were created by using the same weights and network connectivity as the trained architectures, and replacing the ReLU neurons with ASN units — this approach allows us to focus on spike-based coding and for now side steps the question of spike-based learning.

We selected well-known benchmark datasets of increasing complexity to demonstrate the robustness of the presented approach. The IRIS dataset is a classical non-linearly separable 'toy' dataset containing three classes — three types of plants — with fifty instances each, to be classified from four input attributes. Similarly, the SONAR dataset [12] contains 208 entries of sonar signals divided in 60 energy measurements in a particular frequency band, to be classified in metal cylinder or simple rocks classes. Lastly, we use the MNIST dataset [14], which has been a standard testbed for novel image classification methods. It is composed of $60{,}000$ entries of handwritten digits for the training set and $10{,}000$ entries for the validation set.

To carry out classification, for each instance the input neurons receive input current $I(t)$ corresponding to the respective feature values, for a simulation duration of 500 ms. During this period, input neurons generate spikes that are instantaneously transmitted to the next layer. There, the corresponding weighted PSCs are added to the membrane potential $u(t)$ through the smoothing filter $\phi(t)$; note that the smoothing filter effectively causes a delay in signal transmission of order τ_{smooth} per layer. This process is repeated for each successive layer in the network. Output values as used for classification are computed as internal current $I(t)$ in the output neurons, smoothed with longer time constant τ_{rout} for stable performance. At every 1 ms time step t of the simulation, classification performance is computed over all instances of the respective dataset from the outputs $I(t)$ at that time step t. Details for the architecture, training and parameters used are given in a box at the end of this article.

Computing with spikes

For all three datasets and the corresponding four network architectures, we computed the ANN performance and compared that to the ASNN performance. Figure 14 shows classification performance obtained for IRIS, SONAR and MNIST by the various ASNNs as a function of average firing rate in the network (and hence neural coding precision) during classification, obtained by varying the ratio of m_f and ϑ_0. We find that for all benchmarks we achieve performance with the ASNN identical to that of the corresponding ANN once a certain minimum firing rate is used, corresponding to the minimal required neural coding precision in the network. The networks that classify the IRIS and SONAR benchmarks require fairly high firing rates compared to the two MNIST architectures. Since the former architectures are comprised of far fewer neurons as compared to the MNIST networks, this suggests that in such smaller networks the coding precision needs to be quite high.

The different firing rate regimes were obtained by varying the multiplicative factor m_f as a function of ϑ_0, between $0.1\vartheta_0$ and $3\vartheta_0$, with $\vartheta_0 = 0.0128$ for the IRIS dataset, in 30 different simulations. The threshold $\vartheta_0 = 0.0128$ was selected such that the smallest positive input values in the training set were still encoded. For SONAR, we carried out simulations with m_f ranging between $0.1\vartheta_0$ and $3\vartheta_0$, using $\vartheta_0 = 1e^{-4}$. For the MNIST dataset we simulated both an FF-ASNN and C-ASNN architecture. For the FF-ASNN we carried out 35 simulations with m_f ranging between $0.1\vartheta_0$ and $3.5\vartheta_0$, using $\vartheta_0 = 3.9e-3$. For the MNIST networks, compared to the IRIS and SONAR networks, we find that performance is stable over a much greater range of firing rates. For each simulation we computed the time to which 101% of the minimum classification error is reached (Matching Time, MT), e.g., for MNIST-cnn this is when the performance exceeds 99.13%. Given parameters ϑ_0 and m_f, we considered the ASNN network as having performance identical to the corresponding ANN if, in the time window from MT to the end of the simulation (500 ms), the performance stays, on average, above the 101% error threshold. The variance is computed over the same time window, while the firing rate is computed in a time window of 100 ms at the end of the simulation. At low firing rates, the ANN performance is exceeded for some ranges by chance; the high neural coding precision for higher firing rates results in more stable performance, as can be seen in the low variance of the performance on the right part of Figure 14.

For all four ASNNs, we noted both the required minimum firing rate (as set through the ratio of m_f and ϑ_0) to reach the 101% error threshold, and the corresponding simulation time when this performance is first reached. We refer to these values as the Matching Firing Rate (FR) and the Matching Time (MT), and the results are shown in Table 2 in the column 'Lowest FR'. For MNIST, we find that the response time for the FF-ASNN is substantially faster as compared to the C-ASNN. This is likely caused by the fact that the C-ASNN is a deeper network. Additionally, we determined the lowest Matching Time and corresponding Firing Rate (Table 2 in the column 'Lowest MT'). We see that for the large MNIST networks, Matching Time

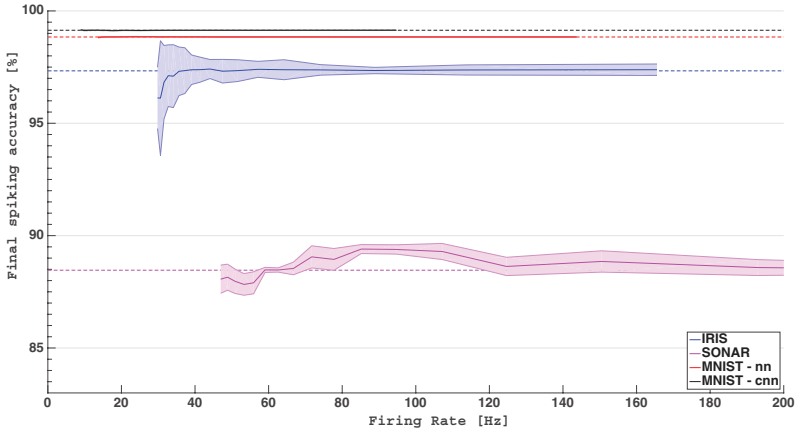


Figure 14 Classification performance on IRIS, SONAR, MNIST (MNIST-nn for FF-ASNN and MNIST-cnn for C-ASNN) for various average firing rates. Dashed: performance of original ANN.

DataSet	ANN	ASNN	Lowest FR		Lowest MT	
		P(%)	FR	MT	FR	MT
IRIS	97.33	97.33	36	107	41.4	46
SONAR	88.46	88.46	59.7	80	77.1	71
MNIST-nn	98.84	98.84	14.6	15	17.3	12
MNIST-cnn	99.14	99.14	8.6	87	10	8.9

Table 2 Performance (%), Matching Firing Rate (FR) (Hz) and Matching Time (MT) (ms).

improves substantially at limited cost in terms of FR. In general, we find that the Matching Time increases with lower firing rates (not shown).

Switching

We also computed the Matching Time to determine that time that input needs to be presented to the network before the output classification reaches ANN performance (101% of the minimum classification error). A more general streaming setting however is one where one stimulus is presented, followed by another stimulus. We illustrate this case in Figure 15: first, white noise is presented to the network for 100 ms, followed by the presentation of a digit, which after 100 ms is then switched to another digit. Shown is the average activation in each layer of the MNIST-cnn ($\vartheta_0 = 3.9e-3$, $m_f = 3\vartheta_0$) for 1000 random stimulus switches, as well as the average activation $S(t)$ in the output neurons and the classification performance. White noise has been reproduced by presenting a (different) Gaussian-noise sampled image with $\mu = 0$ and $\sigma = 0.5\vartheta_0$, at each ms frame. We see that noise only stimulates the first layer, and fails to substantially activate subsequent layers. Once the first actual digit is presented, the network rapidly and correctly recognizes this digit. After 200 ms the permuted images are presented: the classification performance for the new dataset reaches the 101% error threshold after a switching time of $\mathrm{ST} = 186$ ms. This switch from one digit to another is determined by — substantially longer — recovery time due to τ_x. Switching time can be improved by decreasing τ_x, but at the expense of an increase in firing rate.

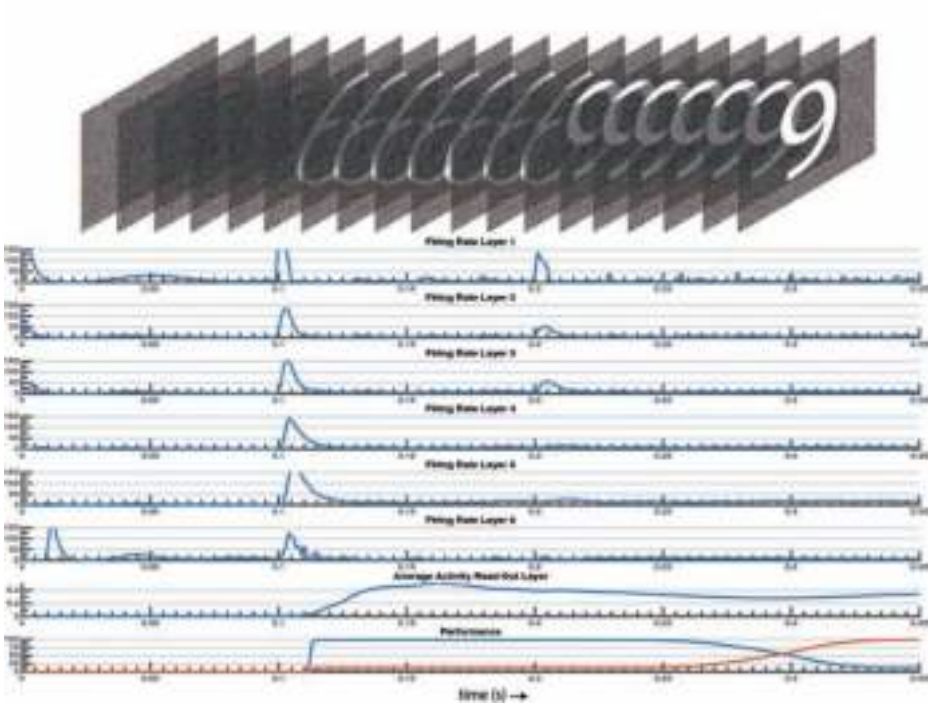


Figure 15 Switching example with C-ASNN. Top: an example of the switching images provided to the network. Middle, rows 1-6: the firing rate of the network's 5 layers plus the read-out layer. Middle, row 7: the average activity of the read-out layer, computed by filtering the internal state of the neurons. Note that, during the noise presentation, although a firing activity in the read-out layer is present, the internal state is silent — a rapid increase in the average activity signals that a classification is made. Bottom: the classification performance through time showing the switch between two test sets of a 1000 digits each.

Discussion and conclusion

We introduced deep neural networks and explained how they are presumed to relate to biology. Given some of the deficiencies of present deep neural networks, we focused on the question of efficient and asynchronous neural coding with spiking neurons.

Spiking neuron models like the ASN presented here capture many important adaptation phenomena in real neurons, and by coupling the synaptic plasticity model, we ensure that downstream neurons appropriately account for adaptation in presynaptic neurons. Thus, it is a prediction of this work that a tight coupling exists between neural adaptation and synaptic plasticity.

At the same time, we demonstrated that the resulting neural network model can replace a standard ANN in a one-to-one manner, without loss of performance, while using an asynchronous and sparse model of spike-based neural computation. As such, the presented ASNN can be considered as a novel paradigm for neural coding with spiking neurons, with an almost direct correspondence to biological spiking neurons.

In particular, we show that the proposed ASNNs can carry out neural computation with performance identical to the corresponding ANN for a number of classical benchmark datasets of increasing network size and complexity. Compared to an otherwise identical SNN that uses Poisson spiking neurons the presented approach has better or identical performance while using a much lower firing rate in the network. Additionally, due to the large dynamic range of the ASNs, no reweighting or normalization of the network was necessary: the ASNs function as drop-in spiking neuron replacements for the ReLU neurons in the standard ANNs. Effectively, the ASN computes using *adaptive* asynchronous sigma-delta pulse modulation, which is necessary because — unlike electrical circuit signals — the signals inside a neural network with ReLU neurons are not bounded to some fixed dynamic range. Note that though we focus here on standard neural networks without recurrence or memory, we recently showed that a similar approach can be applied to networks with memory [20], to learn cognitive tasks, like

tasks that require remembering a value for a number of steps and then being able to act on this value.

Compared to classical ANNs, the computations of the ASNNs are asynchronous, event driven and sparse. To truly exploit the efficiency of sparsely active asynchronous spiking neural networks, efficient GPU or ASIC implementations need to be created. Current CNN implementations are heavily optimized for carrying out convolutions on GPUs, an operation which closely fits the GPUs parallel architecture. For sparsely active neural networks, where most neurons are not active at any given time step, novel approaches need to be developed: since typically for any stimulus only a subset of neurons is active, fast caching methods are likely to hold promise. As most networks of spiking neurons, the reduction in communication between the neurons is traded against more complex dynamics in the neuron; since there are typically orders of magnitude fewer neurons than connections, this trade-off can be worthwhile provided that the neuron model requires limited memory and computation. The ASN model presented here can be computed with only a few variables (principally the components of the γ and η kernels), which when formulated as simple dynamical systems can be computed in a memory-less fashion, without tracking previous spike-times. Emerging hardware architectures like Intel's Loihi chip are eminently suitable for exploiting the efficiency of spike based computation.

Our adapting neurons effectively use analog spikes: each spike is associated with a refractory kernel of different height. In principle, the analog value of a spike can be reconstructed at the postsynaptic neuron from just the time since the previous spike, but at considerable computational expense. Compared to standard (analog) ANNs, the ASNNs compute in an asynchronous and localized manner: input information can be presented to the network at the precision with which neurons are updated, while the rate of information exchange in the network is determined by the neural coding precision required for classification. The network can thus process for instance 1000 Hz input frames when neural updates are carried out with 1 ms time steps: in this manner, new input can be processed almost immediately — albeit with the delay incurred in the consecutive layers. The neural activity is also localized, in that only a subset of neurons is really activated, emitting many spikes, and most neurons are silent or only very sparsely active. Since bandwidth, as used for reading weights from memory, is typically the limiting factor when computing an ANN, the sparse and localized neural computation offers a potentially more efficient way of time-continuous neural computing.

We showed how we can relate spike-based coding to the analog signals that standard artificial neural networks compute with. Such a translation allows us to design sparsely active neural networks while using the existing frameworks (like Tensorflow and PyTorch) to train the analog counterparts of these spiking networks. This is of course sufficient when the aim is to deploy a trained network on low-power hardware. To include learning, we have to develop spike-based learning algorithms. A straightforward solution there is to note that the error that is backpropagated in the error backpropagation learning rule could potentially be carried by a separated network of spiking neurons. Peter O'Connor recently showed some work in that direction [17], but in general the problem with this approach is that the approximation error in the 'spiking' AD/DA conversion becomes too large when the neural networks become very deep. To overcome this, different approaches to learning in deep networks may need to be found, where biology will again be a source of inspiration as most are convinced that the brain does not use error-backpropagation but rather relies on smart and data-efficient forms of learning that learn the natural structure of the world without being given explicit examples, as is needed for error-backpropagation.

Feed-forward neural networks

We trained fully connected FFNNs using dropout [25] to approximately match performance with state-of-the-art. We trained a four layer FFNN of size $[4-30-30-3]$ on the classical IRIS dataset with a dropout rate of 0.5, learning rate of 0.1, for 800 epochs. We used half of the dataset for training, and we obtained 97.33% on the validation set. For the SONAR dataset, we trained a four layer FFNN of size $[60-50-50-2]$, using the training set division reported in [12] for the angle-dependent experiment. We used a dropout rate of 0.5, learning rate of 0.2, and we trained for 1000 epochs to obtain 88.46% accuracy on the validation set. For the MNIST dataset, we used the trained network reported in [7] to directly compare with the method there. In [7], the authors trained a $[784-1200-1200-10]$ network, with a dropout rate of 0.5, learning rate of 1 and momentum of 0.5. With this network, we obtained 98.84% accuracy on the MNIST validation set (code and trained network were available online [27] using a modified version of the DeepLearnToolbox [19, 28]. As in [7], for all datasets the input values were scaled to the range $[0,1]$. We refer to the FFNNs that use ASN ReLU units as feed-forward adaptive spiking neural networks (FF-ASNN).

Convolutional neural networks

CNNs have become a standard tool for image classification tasks [14], and they generally outperform classical FFNNs. In [7] a competitive ReLU CNN implementation for MNIST was presented: we apply the ASN network to this architecture and compare our results to those obtained in [7]. The pre-trained network consists of a $[28\times28-12c5-2s-64c5-2s-10o]$ CNN, where 28×28 corresponds to the input image size, $N,\ c,\ K$ are the N-convolutional kernels of size K, $M,\ s,\ J$ are the M-averaging pooling filters of size J, and o is the size of the output layer; note that this network is available online [27]. Neurons in each of these layers use the ReLu activation function, and we can again map the ANN directly to our ASNN by substituting each ReLU unit with the adaptive spiking neuron. We refer to the CNNs equipped spiking neurons as convolutional adaptive spiking neural networks (C-ASNN).

References

1 Ch. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, 1995.

2 M. Boerlin and S. Denève, Spike-based population coding and working memory, *PLoS Computational Biology* 7(2), February 2011, e1001080.

3 S. M. Bohte, Efficient spike-coding with multiplicative adaptation in a spike response model, in *NIPS* 25, MIT Press, 2012, pp. 1844–1852.

4 S. M. Bohte and J. O. Rombouts, Fractionally predictive spiking neurons, in *NIPS* 23, MIT Press, 2010, pp. 253–261.

5 R. Brette, Spiking models for level-invariant encoding, *Front. in Comp. Neurosc.* 5 (2011).

6 D. B. Chklovskii and D. Soudry, Neuronal spike generation mechanism as an oversampling, noise-shaping a-to-d converter, in *NIPS* 25, MIT Press, 2012, pp. 503–511.

7 P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu and M. Pfeiffer, Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing, *IEEE IJCNN,* July 2015, pp. 1–8.

8 Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko and Trevor Darrell, Long-term recurrent convolutional networks for visual recognition and description, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2625–2634.

9 A. L. Fairhall, G. D. Lewen, W. Bialek and R. R. de Ruyter van Steveninck, Efficiency and ambiguity in an adaptive neural code, *Nature*, 412(6849) (2001), 787–792.

10 W. Gerstner and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity,* Cambridge University Press, 2002.

11 Wulfram Gerstner, A framework for spiking neuron models: The spike response model, *Handbook of Biological Physics*, Volume 4, Elsevier, 2001, pp. 469–516.

12 R. P. Gorman and T. J. Sejnowski, Analysis of hidden units in a layered network trained to classify sonar targets, *Neural Networks* 1(1) (1988), 75–89.

13 Alan L. Hodgkin and Andrew F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, *The Journal of physiology* 117(4) (1952), 500–544.

14 Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE* 86(11) (1998), 2278–2324.

15 Zachary F. Mainen and Terrence J. Sejnowski, Reliability of spike timing in neocortical neurons, *Science* 268(5216) (1995), 1503–1506.

16 Ilya Nemenman, Geoffrey D Lewen, William Bialek and Rob R de Ruyter van Steveninck, Neural coding of natural stimuli: information at submillisecond resolution, *PLoS Computational Biology* 4(3) (2008), e1000025.

17 Peter O'Connor, Efstratios Gavves and Max Welling, Temporally efficient deep learning with spikes, arXiv:1706.04159, 2017.

18 E. Olson, A passive solution to the sensor synchronization problem, in *Int. Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ*, IEEE, 2010, pp. 1059–1064.

19 R. B. Palm, Prediction as a candidate for learning deep hierarchical models of data, 2012.

20 Isabella Pozzi, Roeland Nusselder, Davide Zambrano and Sander Bohté, Gating sensory noise in a spiking subtractive LSTM, submitted, 2018.

21 Ch. Pozzorini, R. Naud, S. Mensi and W. Gerstner. Temporal whitening by power-law adaptation in neocortical neurons, *Nature Neuroscience* 16(7) (2013), 942–948.

22 B. D. Ripley, *Pattern Recognition and Neural Networks*, Clarendon Press, 1996.

23 D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning representations by back-propagating errors, *Nature* 325 (1986), 533–536.

24 L. Ślażyński and S. Bohte, Streaming parallel gpu acceleration of largescale filter-based spiking neural networks, *Network: Computation in Neural Systems* 23(4) (2012), 183–211.

25 N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.* 15(1) (2014), 1929–1958.

26 Y. C. Yoon. LIF and simplified SRM neurons encode signals into spikes via a form of asynchronous pulse sigma-delta modulation, in *IEEE TNNLS*, 2016, pp. 1–14.

27 http://github.com/dannyneispikingJelu_conversion.

28 https://github.com/rasmusbergpalm/DeepLearnToolbox.

Rui M. Castro
Department of Mathematics and Computer Science
Eindhoven University of Technology
rmcastro@tue.nl

Ervin Tánczos
Wisconsin Institute for Discovery
University of Wisconsin–Madison, USA
tanczos@wisc.edu

On adaptive sensing for high-dimensional signal inference

In many practical settings one can sequentially and adaptively guide the collection of future data, based on information extracted from data collected previously. These sequential data collection procedures are known by different names, such as sequential experimental design, active learning or adaptive sensing/sampling. In this paper Rui Castro and Ervin Tánczos give a small overview of recent results characterizing the fundamental limits of adaptive sensing for sparse signal inference. They focus primarily on support estimation for static signals, and detection of dynamically evolving signals. They show that, in many situations, adaptive sensing is able to make reliable inferences in situations where non-adaptive sensing would fail dramatically.

At large, the goal of any statistical learning and inference methodology is to uncover important aspects of a given process (e.g., physical, social or other) by collecting and analyzing data. Naturally, both the collection and analysis of data are essential aspects of the methodology, and as such should not be considered separately. In all but few cases one can view the data collection process as a 'querying' or experimenting procedure, where collected data are 'answers' or outcomes of multiple queries/experiments, frequently corrupted in some fashion.

Often there is significant flexibility in the choice of queries. If this choice must be made before *any* data is collected this corresponds to the classical experiment design scenario. However, if each query/experiment can be chosen sequentially, shaped by the answers/outcomes of the previous queries, much more powerful data collection and inference approaches can be used. Depending on the research area this sensing paradigm is known by different names: *sequential experimental design* in the statistics and economics literature, *active learning* [2, 7, 15] or *adaptive sensing/sampling* in computer science, engineering and machine learning and statistics. An essential aspect of adaptive sensing is the intricate coupling between data analysis and acquisition, which creates a powerful feedback structure. This is a double-edged sword: it is key to harness the power of sequential experimental design but also makes the analysis and design of these procedures rather challenging — indeed it creates complicated and strong dependencies in the data.

More broadly, the feedback between data analysis and acquisition is key in the scientific discovery method, a process of complex interactions between experiments and outcomes, guided mostly by the scientists intuition. There have been a few attempts to semi-automate such processes, notably the efforts reported in [17], where a large team of scientists developed a robot capable of autonomously conducting experiments and test hypothesis leading to the discovery of novel genomic knowledge in yeast cells.

The goal of this paper is to give a small overview of recent results pertaining inference of signals living in high dimensional spaces, as well as an idea of the methods and techniques used in the development of the analysis and algorithms. Although these results pertain only a certain class of settings and problems many of the surrounding methods and ideas are extremely useful in a broader sense.

A sparse signal model

In this paper we survey some recent results pertaining inference of signals *living*

in high dimensional spaces. These signals might be static, meaning they do not change during the measurement period, or dynamic, meaning they evolve while data is being collected. For simplicity, we consider first static signals. Concretely, the signal of interest is represented by an n-dimensional vector $\mathbf{x} \in \mathbb{R}^n$, where n is called the *extrinsic signal dimension*. Depending on the context, the entries of $\mathbf{x}$ might be used to represent different quantities. For instance, in gene expression studies, each entry of the signal vector is associated with a different gene, and the value of the signal is related to the corresponding expression level. Another compelling example pertains the monitoring of computer networks for detection and localization of anomalous behavior. In that case each entry of $\mathbf{x}$ might represent the activity level of each node in the network.

Although in principle the vector $\mathbf{x}$ can be arbitrary, in most cases it is *sparse* meaning most entries of $\mathbf{x}$ have nominal/typical values, and only relatively few entries have values deviating from that norm. Referring back to the two examples above most genes will be expressed to their nominal value, and only a few genes (e.g., about 100 out of 20.000 genes) will have a different expression level, for instance, due to the onset of a disease. In the case of network monitoring most nodes in the network will behave normally, and only a few nodes will have higher activity, for instance, if those nodes are participating in a distributed denial-of-service attack.

To abstract this assumption let S be a subset of $\{1,\ldots,n\}$ of non-zero entries of $\mathbf{x}$ and assume that for all $i \in \{1,\ldots,n\}$ such that $i \notin S$ we have $x_i = 0$ (the nominal value of each entry). We refer to S as the *signal support* and this is our main object of interest. We might want to estimate the signal support set, or simply detect if S is not the empty set, as we formalize below. In the section 'Dynamically evolving signals' we consider also a modification of this model to allow signals to evolve over time.

To greatly simplify the presentation in this paper we consider only signals of the form

$$x_i = \begin{cases} \mu & \text{if } i \in S, \\ 0 & \text{if } i \notin S, \end{cases}$$

where $\mu > 0$ is called the *signal amplitude*. This restriction is also considered in [1, 11] in the non-adaptive sensing context and does not substantially hinder the generality of the results presented in this manuscript. In particular, when characterizing the difficulty of the problem in terms of the minimum signal magnitude $\min_i |x_i|$ the characterization we provide is essentially sharp. Throughout this paper we consider the sparse regime where $|S| \ll n$ (meaning $|S|$ is much smaller than n). More specifically we assume $|S| \leq n/\log_2(n)$.

Measurement model

Naturally, we do not have access to the signal $\mathbf{x}$ directly. Rather, we can collect only partial information through noisy measurements of individual entries, or possible ensembles of the signal entries. The latter is often referred to as Compressive Sensing (see [9, 12] and references therein). In this paper we focus primarily on the first model. This sensing model was introduced in [16]. We assume it is possible to collect measurements of each signal component corrupted by additive Gaussian noise. Concretely let

$$Y_k = x_{A_k} + \Gamma_k^{-1/2} W_k, \quad k = 1, 2, \ldots,$$

where k denotes the measurement index (first measurement, second measurement, and so on), A_k denotes the entry of $\mathbf{x}$ being measured, Γ_k denotes the corresponding *precision* of the measurement, and W_k is a standard normal random variable embodying measurement noise. Importantly, W_k are independent of $\{Y_i\}_{i=1}^{k-1}$ and also independent of $\{A_i, \Gamma_i\}_{i=1}^{k}$. Said differently, each measurement corresponds to a single signal entry corrupted with additive Gaussian noise. Both choices of which entry to measure and the corresponding noise level can be determined by experimenter — the higher the precision of a measurement, the lower the noise level. Note also that it is possible to measure the same signal component multiple times, with independent noise realizations.

However, it is not possible to measure all the entries with arbitrarily large precision. In particularly, there is a total sensing budget constraint that must be satisfied, namely

$$\sum_{k=1}^{\infty} \mathbb{E}(\Gamma_k) \leq m, \tag{1}$$

where $m > 0$. This means that it is not possible to measure all the signal entries with very high precision. Note that the above constraint is on the *expected* precision used. Alternatively, we can consider a slightly more stringent constraint on the actual precision (i.e., $\sum_{k=1}^{\infty} \Gamma_k \leq m$). The formulation in equation (1) considerably simplifies the presentation, but does not qualitatively alter any of the results presented below. The reason is that, for most reasonable algorithms, control over the expected precision translates into control over the actual total precision in high probability by a concentration of measure argument.

This model might seem peculiar at first, as it allows for a scenario where one takes an infinite but countable number of measurements (provided the precision decays sufficiently fast as a function of k). Nevertheless, this model fits closely several practical situations, namely when the precision of measurements is proportional to the time it takes to collect a measurement. In that case equation (1) is stating one has to take measurements during a time period with the duration of m units. A representative example of such a setting is in astronomical surveys, where long exposure times are used to reduce the noise level. It is also possible to consider instead settings where there is little or no control over the precision (e.g., say $\Gamma_k \equiv \Gamma$ for all $k \in \{1, 2, \ldots\}$). In that case the constraint in equation (1) simply states there is a maximum number of measurements that are allowed. This situation is closely related to what is encountered in stochastic multi-armed bandits settings [3].

Allowing for arbitrary precision values has several advantages: it is a more general measurement model, and allows for cleaner analytic results. Nevertheless, the entire methodology and analysis can still be done when considering fixed precision measurements and a constraint on the total number of measurements. When discussing inference of dynamically evolving signals in the section 'Dynamically evolving signals' we actually consider that situation instead.

The measurement model above is rather general, and allows for adaptive sensing approaches. Namely, one can adjust the way new measurements are collected based on measurements collected earlier. In other words, one can choose A_k, Γ_k as a function of the past experiments $\{Y_i, A_i, \Gamma_i\}_{i=1}^{k-1}$. Furthermore, this choice can also incorporate extra randomness, if desired. The collec-

tion of conditional distributions of A_k, Γ_k given $\{Y_i, A_i, \Gamma_i\}_{i=1}^{k-1}$ for all k is referred to as the *sensing strategy*. We can also consider more traditional non-adaptive sensing strategies. In that case the choice of sensing actions and corresponding precision must be made before collecting any data. Formally this means that $\{A_k, \Gamma_k\}_{k\in\mathbb{N}}$ is statistically independent from $\{Y_k\}_{k\in\mathbb{N}}$. Note that a non-adaptive design can still be random.

The case $m = n$ is of particular interest, allowing for a simple direct comparison between adaptive and non-adaptive sensing methodologies. When $m = n$ we allow on average one unit of precision per each of the signal entries. So, if there is no reason to give preference to any particular entry of $\mathbf{x}$, the natural optimal non-adaptive sensing strategy should simply measure each entry of $\mathbf{x}$ exactly once, with precision one. This corresponds to the well studied normal means model.

Although we are considering measurements with additive Gaussian noise, the methodologies used to study inference problems in this setting are easily extended to more general distributions (see for instance [20]). Actually, the setting described here is closely related to a class of problems known as *stochastic multi-armed bandits* (see [3] for an excellent survey). In North-American slang, a *one-armed bandit* is a casino slot-machine. In the stochastic multi-armed bandit setting one can imagine a row of n slot machines. Each machine is endowed with an unknown probability distribution F_i with mean x_i, $i \in \{1, \ldots, n\}$. At each turn k the experimenter can choose a machine A_k to play, and will observe a 'reward', nothing more than a sample of F_{A_k}. Depending on the setting the experimenter might have different goals. He/she might want to maximize the total reward — this leads to an exploration/exploitation trade-off. Or instead they might want to identify the best-paying machine, leading to the so called pure-exploration problem. The latter is intimately related to the problem of signal detection described above, which has essentially the same statistical complexity as locating a single non-zero signal component. Actually, in [13] a methodology for proving lower bounds for the best-arm problem was developed, which is essentially the same methodology developed earlier for the detection problem [5].

Support estimation

As stated earlier, we focus mainly on two classes of inference problems — support estimation and signal detection. We start by addressing the problem of support estimation, in which the goal is to identify the signal support S. In other words, one desires to construct a set estimator $\hat{S}$, based on the data $\{A_k, \Gamma_k, Y_k\}_{k=1}^{\infty}$ that is 'close' to the true signal support S. There are different ways to measure the closeness of these two sets. For concreteness we focus on number of errors, which is given by the cardinality of the symmetric set difference $\hat{S}\Delta S = (S \backslash \hat{S}) \cup (\hat{S} \backslash S)$.

Let us assume $S \in C$, where C is a given class of non-empty subsets of $\{1, \ldots, n\}$. For simplicity of presentation we assume that all the sets in C have the same cardinality s. A prototypical example is the class of all support sets with cardinality s (consisting of $\binom{n}{s}$ sets). The goal of support set estimation is therefore to construct a support set estimator $\hat{S} \equiv \hat{S}(\{A_k, \Gamma_k, Y_k\}_{k=1}^{\infty})$ such that worst case error

$$\max_{S\in C} \mathbb{E}_S[\hat{S}\Delta S]$$

is small. In the above $\mathbb{E}_S$ denotes the joint probability distribution of $\{A_i, \Gamma_i, Y_i\}_{i=1}^{\infty}$ for a given support set S.

For support estimation other metrics can be considered, such as $\mathbb{P}(\hat{S} \neq S)$ or False Discovery Rate (FDR) plus Non-Discovery Rate (NDR). The first is very similar to what is described above while the second metric is more lenient: we only try to make the number of errors *relative to* the size of $\hat{S}$ and S small. This means weaker signals can be reliably recovered. See for instance [5, 16].

Non-adaptive sensing

Provided the class C of possible support sets is somewhat symmetric, there is no a priori reason to measure any given signal component in detriment to another component. The formal definition of symmetric classes is given below.

Definition 1. Let $S \in C$ be drawn uniformly at random. If $\mathbb{P}(i \in S)$ has the same value for all $i = 1, \ldots, n$, the class is said to be *symmetric*.

Equivalently, if $\sum_{S\in C} \mathbb{1}\{i \in S\}$ is not a function of i, the class is said to be symmetric. In the previous expression $\mathbb{1}$ denotes the usual indicator function.

For symmetric classes there is no reason to measure any signal component with more precision than any other. Therefore, if considering the non-adaptive sensing paradigm, the only reasonable sensing strategy is to use *uniform sensing*. This means that we distribute our sensing budget uniformly over all the signal components. Furthermore, collecting more than one measurement per signal entry is not necessary, due to statistical sufficiency. Therefore the optimal non-adaptive sensing strategy consists of n measurements (one per signal components), each with precision m/n. Said differently, we collect n independent measurements $Y_i \sim \mathcal{N}(x_i, n/m)$, $i \in \{1, \ldots, n\}$.

Since this is an estimation problem it is sensible to consider a maximum likelihood approach. Namely, construct $\hat{S}$ by choosing the support set S that maximizes the likelihood of the observations. It is not hard to show that this gives rise to the estimator

$$\hat{S}_{\text{non-adaptive}} = \operatorname*{argmax}_{S\in C} \sum_{i\in S} Y_i. \qquad (2)$$

When considering the class of all support sets of cardinality s this methodology simply deems the s largest observations as the support estimate. Naturally, to ensure the expected number of errors is small the signal magnitude μ must be above the 'noise floor'. In particular for an arbitrary $\varepsilon > 0$, if $\mu \geq \sqrt{\frac{2n}{m}(1+\varepsilon)\log n}$ then it is guaranteed that

$$\max_{S\in C} \mathbb{E}_S[\hat{S}_{\text{non-adaptive}}\Delta S] \to 0,$$

as $n \to \infty$. Conversely, if $\mu < \sqrt{\frac{2n}{m}(1-\varepsilon)\log n}$ such a methodology will necessarily fail. In fact, one can show that if $\mu < \sqrt{\frac{cn}{m}\log\frac{n}{s}}$ for c slightly smaller than $\frac{1}{2}$, then no method whatsoever will be able to reliably estimate the support. More generally it turns out that under mild conditions the cardinality of the class C is the main bottleneck in regards estimation of the signal support. The following result characterizes the limits of *any* support estimation procedure.

Proposition 1. (Proposition 10 of [8]) *Suppose that C is symmetric, it only contains sets of size s and that $1+\sqrt{2} \leq (1-2\varepsilon)\log(|C|-1)$, where $|C|$ denotes the cardinality of C. If*

$$\mu \leq \sqrt{(1-2\varepsilon)\frac{n}{2sm}\log(|C|-1)},$$

then no non-adaptive procedure can satisfy

$$\mathbb{P}(\hat{S} \neq S) \leq \varepsilon.$$

Proposition 1 deals with a different error metric than the one we considered before. Note however that $\mathbb{P}(\hat{S} \neq S) \leq \mathbb{E}[|\hat{S} \Delta S|]$, hence the proposition also holds with $\mathbb{P}(\hat{S} \neq S)$ replaced by $\mathbb{E}[|\hat{S} \Delta S|]$ in the statement.

For the class of all sets of cardinality s we have $|C| = \binom{n}{s} \approx (n/s)^s$, giving rise to the claim made earlier. In a nutshell, in order for non-adaptive sensing to be successful in recovering a sparse signal the signal magnitude needs to scale roughly like $\sqrt{\frac{n}{m} \log n}$ (since when $s \ll n$, $\log \frac{n}{s}$ has the same scaling as $\log n$). If the class C has some structure (so the support sets are not arbitrary subsets of $\{1,\ldots,n\}$) slightly better results can be obtained (see [8] and the subsection 'Structured support estimation').

Adaptive sensing

Having established that it is not possible to recover the support of signals with non-adaptive sensing when magnitude of the active components is roughly smaller than $\sqrt{\frac{n}{m} \log n}$, a natural question to ask is if it is possible to do better with adaptive sensing. As the reader might expect, the answer is affirmative, and we describe next a simple approach and analysis demonstrating this.

Consider a procedure in which we test each entry $x_i, i \in \{1,\ldots,n\}$ independently to assess if these are zero or not. To perform the test, for entry i we take repeated measurements with the same precision Γ. If a negative measurement is collected we interpret that as evidence supporting the case that $x_i = 0$; hence we terminate the test and decide $i \notin \hat{S}$. On the other hand, if we measure a component T times without ever seeing a negative value, we interpret it as evidence supporting $x_i = \mu > 0$; we terminate the test and decide $i \in \hat{S}$.

Remarkably, this simple procedure performs very well — in fact almost as well as the best possible procedure — as we illustrate in the analysis below. Note that we still need to specify the parameters Γ and T. A good choice will become apparent from the analysis.

To evaluate the performance of this method we need to do two things. On one hand we need to assess the error that this method incurs. On the other hand, we need to ensure the precision budget of equation (1) is not exceeded. We start with the latter.

Since all measurements are made with the same precision Γ, our task is to count the total number of measurements the procedure makes in expectation. The expectation of the total number of measurements is simply the sum of the expected number of measurements for each individual test. Note that there are $(n-s)$ zero components in x. In this case, the probability that we observe a negative value is $\frac{1}{2}$. Since we stop measuring a component once we see a negative value, the expected number of measurements in such cases is 2. On the other hand, the test never performs more than T measurements for any coordinate. Hence the expected precision used by the test can be upper bounded as

$$\mathbb{E}\left(\sum_k \Gamma_k\right) \leq \Gamma(2(n-s) + sT). \qquad (3)$$

This can be used to determine the precision Γ of each measurement once we determined the adequate choice for T.

The next step is to control the number of errors. Since the procedure tests each component independently, this means we only need to understand the probability of error for a single test. Consider the test of component i, where $i \in \{1,\ldots,n\}$. In case $x_i = 0$, an error is made if we observe T consecutive positive measurements. Since the probability of any one measurement being positive is $\frac{1}{2}$, the probability of making an error (meaning $i \in \hat{S}$) is simply 2^{-T}.

In case $x_i = \mu$, an error is made unless all measurements are positive. Let Φ denote the cumulative distribution function of a standard normal distribution (specifically, if $Z \sim \mathcal{N}(0,1)$ is a standard normal random variable, then $\Phi(z) = \mathbb{P}(Z \leq z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$ for $z \in \mathbb{R}$). Denoting the (possible) measurements by $Y_{i,1},\ldots,Y_{i,T}$ we see that the error probability is upper bounded by

$$\begin{aligned} &\mathbb{P}(\exists j \in \{1,\ldots,T\} : Y_{i,j} < 0) \\ &\leq \sum_{j=1}^{T} \mathbb{P}(Y_{i,j} < 0) \\ &= T\Phi(-\sqrt{\Gamma}\,\mu) \\ &\leq \frac{T}{2} \exp(-\Gamma\mu^2/2), \end{aligned}$$

where $Y_{i,j} \sim \mathcal{N}(\mu, 1/\Gamma)\ \forall j \in \{1,\ldots,T\}$, and we used a union bound and a standard bound for the tail of a Gaussian distribution (for any $z \geq 0$ we have $1 - \Phi(z) = \Phi(-z) \leq \frac{1}{2} e^{-z^2/2}$). Putting all this together, we get that number of errors of the procedure can be upper bounded as

$$\begin{aligned} \mathbb{E}_S[\hat{S} \Delta S] &= \sum_{i \in S} \mathbb{P}(i \notin \hat{S}) + \sum_{i \notin S} \mathbb{P}(i \in \hat{S}) \qquad (4) \\ &\leq \frac{sT}{2} \exp(-\Gamma\mu^2/2) + (n-s)\, 2^{-T}. \end{aligned}$$

Now all that we have left is to choose the parameters Γ and T such that equation (1) is satisfied and that the error is small, say smaller than a pre-determined level $\varepsilon > 0$. Note that this is a balancing act: on one hand we would like to choose the precision Γ and the maximal number of measurements T large, so that the error becomes small, as shown by inequality (4). However, doing so increases the amount of precision used by the procedure, as shown by inequality (3).

Note that the second term on the right hand side of inequality (4) only involves T. Hence, if we want the error to be at most ε than T has to be chosen to make that term strictly smaller than ε, say $\varepsilon/2$. This leads to the choice $T = \log_2(2(n-s)/\varepsilon)$. However, once we have T chosen, Γ is determined by the need to satisfy the precision budget. In particular, using the choice $\Gamma = \frac{m}{3n}$ with inequality (3) yields

$$\mathbb{E}\left(\sum_k \Gamma_k\right) \leq m\left(\frac{2(n-s)}{3n} + \frac{s \log_2(2(n-s)/\varepsilon)}{3n}\right).$$

Note that the first term in the brackets above is at most $\frac{2}{3}$. The second term in the brackets is a function of n, s and ε. However, we are considering scenarios where n is very large, and s is much smaller than n. In such cases, unless ε is chosen to be extremely small, this term is at most $\frac{1}{3}$. Stated differently, for a given $\varepsilon > 0$ and assuming $s \leq n/(\log_2(n))$ equation (1) is satisfied provided n is large enough.

The last piece of the puzzle is to figure out how large the signal strength μ needs to be so that the error of the procedure above is at most ε. We have seen that our choice of T ensures that we do not erroneously include zero components in $\hat{S}$, but unless the signal is strong enough we cannot expect to correctly identify non-zero components. Formally, we see from inequality (4) that to ensure an error of at most ε we need

$$\frac{sT}{2} \exp(-\Gamma\mu^2/2) \leq \frac{\varepsilon}{2}.$$

Rearranging the above, and plugging in values for T and Γ yields

$$\mu \geq \sqrt{\frac{6n}{m}\left(\log \frac{s}{\varepsilon} + \log\log_2 \frac{2(n-s)}{\varepsilon}\right)}.$$

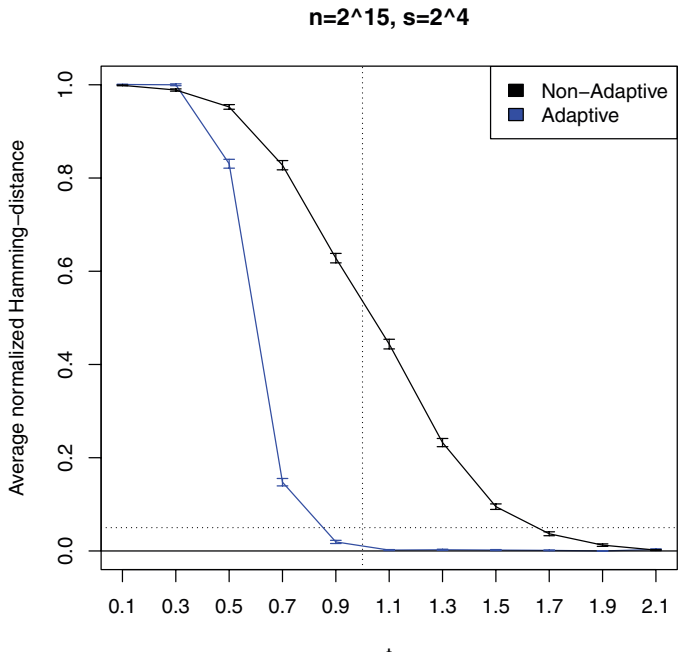


Figure 1 Average normalized error (with SE bands) for the different estimators as a function of the parameter t (the signal strength is $t \cdot \mu_{\text{limit}}$): the non-adaptive estimator(black); adaptive sensing estimator (blue). The number of repetitions is 100 for each value of t. The vertical black dashed line is at the value $t = 1$. The horizontal black dashed line is at the value of $\varepsilon = 0.05$.

In other words we have shown that whenever the signal strength satisfies the previous inequality, the procedure described above has error at most ε.

A few remarks are now in order. We first remark that the $\log\log n$ term in the bound above is an artifact of the simple method we presented, and one can use a more refined procedure to eliminate it from the bound. The same is true for the constant 6 in the bound, which can be improved to a value of 2. Actually, for each coordinate, the procedure described is essentially a poor-man's version of the celebrated Sequential Likelihood Ratio (SLR) test [24]. The latter is significantly more involved, but its performance is qualitatively the same.

To summarize, there is a slightly more sophisticated procedure which can achieve error no larger than ε provided that $\mu \geq \sqrt{\frac{2n}{m}\log\frac{s}{\varepsilon}}$. Contrasting this with the non-adaptive bounds presented before, we see that we can improve the $\log n$ factor to a $\log s$ factor in the bounds for μ using an adaptive procedure. In particular, this adaptive procedure is able to identify supports of signals that are so weak that doing so with any non-adaptive procedure is completely hopeless.

In Figure 1 we give a comparison of non-adaptive and adaptive sensing based on simulation. We plot the performance of the adaptive and non-adaptive procedures for different values of the signal strength μ. Theory tells us that the error of the adaptive procedure should drop below the value ε when $\mu = \mu_{\text{limit}} := \sqrt{\frac{2n}{m}\log\frac{s}{\varepsilon}}$. Hence we run the procedures for signal strength $\mu = t \cdot \mu_{\text{limit}}$ with values of t varying around 1. Our other parameter choices are $n = 2^{15}$, $m = n$ and $s = 2^4$. For each value of t we run both methods 100 times and plot their average error along with error bars whose total length is four times the (point-wise) standard error, which would correspond to a roughly 95% two-sided confidence interval for normally distributed measurements. Finally we note that instead of the simple procedure described in this manuscript, we use the somewhat more sophisticated test described in [8] that replaces the simple thresholding procedure with SLR tests, as described above. However, the results do not differ substantially when using the simple thresholding procedure we described. As can be seen in Figure 1 adaptive sensing clearly outperforms non-adaptive sensing, for the same precision budget, as expected.

Structured support estimation

We have seen that adaptive support estimation procedures have a marked advantage over non-adaptive procedures. In particular, when $m = n$ adaptive procedures can identify the support of sparse signals whose strength scales as $\sqrt{\log s}$, whereas the weakest sparse signal any non-adaptive procedure can deal with needs to scale as $\sqrt{\log n}$. The gap between the performance of adaptive and non-adaptive procedures can even be more dramatic, if the support of the signal is known to have some structural properties.

Assuming the signal support has some structure is a reasonable assumption in many applications. For instance, in gene expression studies one knows the expression levels of certain genes tend to be correlated, giving rise to interval-like structures in the gene-expression vector. In the same context, if one stacks the gene-expression levels of different individuals into a matrix, one expects to see that individuals with the same phenotype (i.e. having the same medical condition) have similar genes expressed, giving rise to sub-matrix structures in the gene-expression matrix. As another example, while monitoring infections on a network (be it a biological infection over a geographic region or the spreading of malware on a network of computers) we may expect to see star-shaped patterns of anomalous behavior radiating from the point of origin of the infection.

Structural assumptions are naturally incorporated in the model by restricting the class $\mathcal{C}$ to contain only sets with specific structural properties. For instance, when thinking of interval structures in signal vectors, we might restrict $\mathcal{C}$ to contain only sets that consist of s consecutive elements of the vector, instead of every possible set with cardinality s as done earlier. Similarly, if we would like to consider star-shaped activations of size s in a network, we would simply restrict $\mathcal{C}$ to contain sets of size s corresponding to star-shaped patterns. More precisely, suppose there is a graph $G = (V,E)$ encoding our network, and we want to consider star-shaped edge activations in G. Then we simply construct a map $\psi : E \to \{1,\ldots,n\}$ that identifies each edge of the network with a number between 1 and n. Then $\mathcal{C}$ consists of elements $\phi(S)$, where S is a star-shaped edge pattern in the network graph.

Under such assumptions, one might want to tailor their support estimation procedure to be more sensitive to such activation patterns, hopefully being able to identify even weaker signals. For instance

we might imagine that we would scan the signal for activations of a specific structural form only. The hope is that since we are considering only a restricted set of activation patterns, we might be able to 'focus' our attention better, resulting in improved performance.

Non-adaptive sensing. We can naturally adapt non-adaptive procedures to cope with structural assumptions. We are primarily interested in situations where, a priori, no component of the signal is more likely to be active. This is formally stated as the assumption that the class C is symmetric, as in Definition 1. As discussed before, for symmetric classes there is no reason to measure any signal component with more precision than any other. In other words, the optimal non-adaptive sensing scheme still measures every component once, with the same precision m/n, and the non-adaptive estimator also remains unchanged and is given by (2). However, in some situations the set C under consideration is much smaller than before, which can benefit the performance of the estimator, as illustrated by Proposition 1.

Considering the class of intervals of size s, the size of the class is $|C| \approx \frac{n}{s}$ and the bound becomes $\sqrt{\log \frac{n}{s}/s}$. This is a significant improvement over the case of unstructured supports. This arises from the fact that the class of intervals of size s is much smaller than the class of all supports of size s. Unfortunately, the possibility for improvement vanishes if the class under consideration contains too many sets, even if the sets themselves have considerable structure to them. An example of this is the class of star-shaped activations in a network modeled by a complete graph. The number of star-shaped edge patterns in a complete graph $G=(V,E)$ is $|C| = |V| \binom{|V|-1}{s} \approx (\sqrt{n}/s)^s$, since we have $|V|$ choices for the center of the star and $\binom{|V|-1}{s}$ choices for the edges, given the center. This results in the bound of the order $\sqrt{\log \frac{n}{s}}$ which is the same as the bound for the unstructured case. In words, even though there is considerable structure to the sets in this class non-adaptive are not able to capitalize on that.

Adaptive sensing. In stark contrast with the above observations is the performance of adaptive sensing procedures. It turns out that the cardinality of the class C no longer plays the crucial role in the performance. A completely general characterization of the performance of adaptive procedures for structured classes is still missing, but we nevertheless have a general way to approach the problem.

A way to construct adaptive sensing procedures is to first consider the problem without observation noise. In such a case a reasonable approach is to first search across the signal vector until we find an element of the support. Once that happens, we can transition into an exploitation phase to find 'nearby' elements of S, taking advantage of any local structure the support might have. Once the local structure has been exploited, the algorithm can revert back to the exploration phase to find another signal component, if there is a part of S that has not been found yet.

Once we have a method for noiseless observations, we can make it robust to noise by repeating each 'noiseless observation' with hypothesis tests to determine the identity of the component in question. These tests can be very similar to the one described previously, or simply SLR tests. In order to get a reliable procedure, the tests need to be properly calibrated — a full description of the approach is too involved to be presented here, but it is fully outlined in [8].

As an example, consider a simple procedure for the case of interval activations. This procedure will consist of two phases. The first phase will be designed to find one component of the signal support S. We call this the exploration phase. Once a component of S has been found, we move on to the exploitation phase, in which we find the remaining elements of S by sampling in the vicinity of the previously found component.

It can be shown that, for the class of intervals with $s < \sqrt{n}$, such a procedure has error smaller than ϵ provided $\mu \geq \sqrt{\frac{2}{s}\log\frac{2s}{\epsilon}}$. Conversely, recovering the support is impossible for weaker signals. So, for the class of intervals adaptive sensing has a gain over adaptive sensing by replacing the factor $\sqrt{\log n}$ by $\sqrt{\log s}$. This is relatively modest.

Now consider the class of star-shaped sets. In that case, one can show that an adaptive sensing procedure (very similar to the one above) has error smaller than ϵ provided $\mu \geq \sqrt{\frac{2}{s}\log\frac{4s}{\epsilon}}$ (this result assumes s slightly smaller than $\sqrt{n}$). In contrast any non-adaptive procedure needs the signal magnitude to have order $\sqrt{\log \frac{n}{s}}$, which is dramatically higher and essentially the same performance had we not taken structure into consideration. In summary, adaptive sensing is able to capitalize on structural assumptions to a much greater extent than non-adaptive sensing.

In the next section we encounter a similar phenomenon, but in a different context — that of signals that change during the measurement period.

Dynamically evolving signals

In the previous sections we considered situations where the observed phenomenon did not change during the measurement process. However, in certain applications this assumption is not completely realistic. Consider for instance a signal intelligence setting where one wishes to detect covert communications. Suppose that our task is to survey a signal spectrum, a small fraction of which may be used for communication, meaning that some frequencies would exhibit increased power. On one hand we do not know beforehand which frequencies are used, but also the other parties may change the frequencies they communicate through over time. This means we will be chasing a moving target. This introduces a further hindrance in our ability to detect whether someone is using the surveyed signal spectrum for covert communications. Other motivating examples for such a problem include spectrum scanning in a cognitive radio system [4, 18], detection of hot spots of a rapidly spreading disease [19, 22, 25, 26], detection of momentary astronomical events [23] or intrusions into computer systems [14, 21].

Signal model

To investigate such settings we first need to develop a model that captures the dynamical nature of such signals. As a start, our aim is to create a model that resembles the previous one as much as possible, so that we can clearly isolate the effect the dynamics has on the problem. Let $x^{(t)}$ denote the signal at time t, where $t = 1, 2, \ldots$. At each time step, the signal is of the form

$$x_i = \begin{cases} \mu & \text{if } i \in S^{(t)}, \\ 0 & \text{if } i \notin S^{(t)}, \end{cases}$$

where $S^{(t)}$ is the support of the signal at time t. The support is always a set of

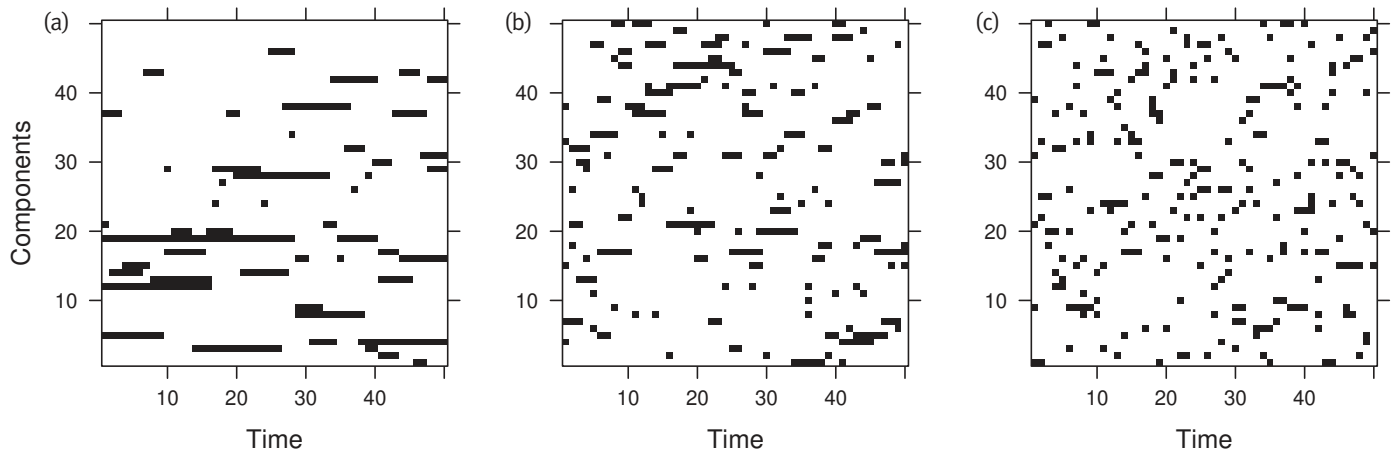


Figure 2 Simulation of the support dynamics with $n = 50$, $s = 5$ and (a) $p = 0.2$; (b) $p = 0.5$; and (c) $p = 0.8$. Components in the support are colored black.

cardinality s as before, but the identity of the elements of the support changes over time. To initialize, $S^{(1)}$ is chosen uniformly at random among all sets of size s. For later times $t = 2, 3, \ldots$, in order to get the support $S^{(t)}$ from $S^{(t-1)}$, we flip a coin for each element of $S^{(t-1)}$ (a total of s flips). If the coin comes up heads, that element will also be included in $S^{(t)}$. On the other hand, if the coin comes up tails, that element 'moves' to a location chosen uniformly at random among the available locations (all those not corresponding to head-flips). The coins we flip comes up tails with probability p.

In this setup p dictates the speed of change of the signal. When $p = 0$, the coin always comes up heads, and thus components never move, or in other words $S^{(t)} = S^{(t+1)}$ for all times t. In other words, the case $p = 0$ is the same as the unstructured model of the previous section. On the other extreme, when $p = 1$ the coin always comes up tails, therefore all components of the support 'move' at each time step. That is to say, when $p = 1$ a new signal support is drawn uniformly at random at each time step. This dynamics are illustrated in Figure 2.

Measurement model and inference goal

As before, the signal cannot be observed directly, only through some sort of noisy measurement mechanism. We consider the same measurement model as before, but with the constraint that the precision of each measurement is 1. This is aligned with the view that the precision of a measurement is proportional to the time it takes to collect that measurement, as already mentioned in the section 'Measurement model'. Formally, the measurement model is

$$Y_t = x^{(t)}_{A_t} + W_t, \quad t = 1, \ldots, m, \tag{5}$$

where A_t denotes the component of $x^{(t)}$ we measure at time t, and W_t is independent standard normal noise. Note that we are allowed to make a total of m measurements, which is a direct analogue of the precision budget constraint of equation (1).

Since the signal support might be changing between time steps, it is not clear what we would mean by support estimation in this setting. However, there is a closely related question one often asks in the context of sparse signals, which is well-defined even for dynamically evolving signals — this is the task of signal detection. Recall that support estimation equates to identifying components of the signal that exhibit anomalous activity. A statistically easier question is whether there is any anomalous activity at all? This can be formalized as a test between two hypotheses. Under the null, every component of the signal behaves nominally, or in other words $S^{(t)} \equiv \emptyset$ for every $t = 1, \ldots, m$. Under the alternative, there is in fact anomalous activity in the signal, evolving according to the model described above. Our task is to decide which of these two hypotheses does our data support more.

Common sense tells us that unless we have enough time to monitor the system, there is no hope of reliably deciding between the two hypothesis, regardless of the signal strength μ. The reason is simple: since the support is sparse, there is very little chance at any given time of sampling an element of the support (under the alternative). To illustrate, imagine a situation where there is no measurement noise. In this case, the reasonable thing to do would be to collect samples of randomly selected components, until we find one whose value is non-zero. This would be hard evidence for the alternative hypothesis. If we sample for a long time, and never came across a non-zero entry, that would be evidence for the null. But how long is long enough? One can show that unless the number of measurements is of the order n/s, there is no hope for any procedure to reliably solve the signal detection problem, regardless of the signal strength μ or the speed of change p. Because of this, we call the setting when $m \approx n/s$ the *small sample regime* and this is the setup we focus on for now. If we are interested in a situation where our goal is to make a decision as fast as possible, this is the setting to consider.

Overview of results

We would like to understand how non-adaptive and adaptive sensing methods fare in the signal detection task described above. Recall that in non-adaptive sensing, the decision which components to measure needs to be made before the sampling process begins. It turns out that, in the small sample regime, static and fast-moving signals are equally hard to detect. One can formalize a necessary condition for the signal strength that needs to be satisfied for *any* non-adaptive method to work. Due to technical difficulties, a formal proof is so far only available in the extreme cases when $p = 0$ and $p = 1$. In both cases,

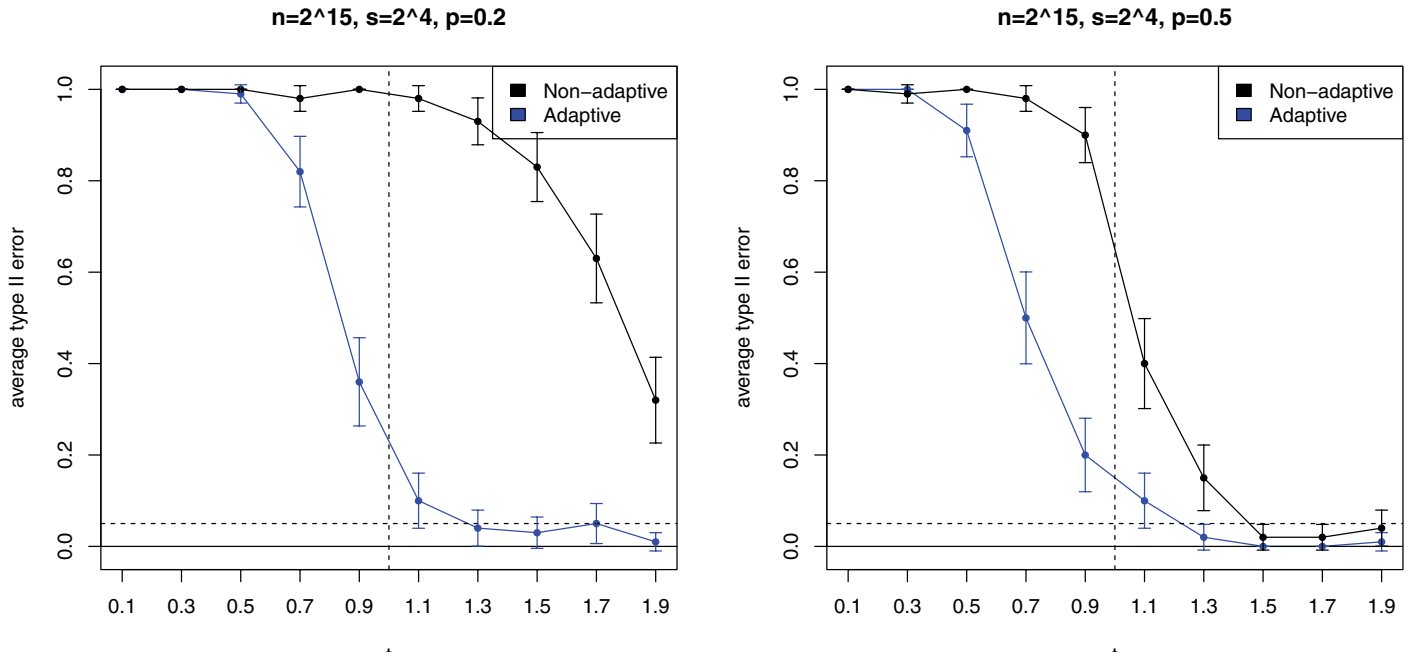


Figure 3 Average type II error probabilities (with SE bands) for the different estimators as a function of the parameter t (the signal strength is $t \cdot \mu_{\text{limit}}$ with $\mu_{\text{limit}} = \sqrt{4p\log(n/s)}$): the non-adaptive test (black); the adaptive sensing test based on the STT (blue). The plots from left to right correspond to $p = 0.2$ and $p = 0.5$. The number of repetitions is 100 for each value of t. The vertical black dashed line is at the value $t = 1$. The horizontal black dashed line is at the value of $\varepsilon = 0.05$.

the signal needs to be at least of order $\sqrt{\log \frac{n}{s}}$ for reliable detection to be possible by non-adaptive methods (see Theorem 4.1 of [10]). This shows that indeed static signals are as difficult to detect as constantly changing ones. Although a formal proof is missing, common sense leads one to believe that the detection boundary should follow the same scaling for $0 < p < 1$ as well.

We now turn our attention to adaptive sensing methods. As before, our hope is that the flexibility in the sampling process would translate to improved performance over non-adaptive methods. In particular, perhaps there is a way to apply a similar strategy to the one we saw before, i.e. to quickly discard zero components and focus sensing effort on non-zero components. Indeed this can be done, using a test with similarities to the one described earlier. Note, however, that we need also to take into account the signal is dynamic, which means that a non-zero component might move away while we are collecting samples of it. Therefore, we need to make our decisions fast, while also keeping our accuracy high. This can be achieved by subtle changes to the sequential test described earlier — we refer the interested reader to [10].

Once we have a test that can quickly and reliably identify whether a signal component is zero or not, the detection procedure is quite simple. Our strategy is to probe roughly $\frac{n}{s}$ components of the signal one after the other. When probing a component, we take repeated measurements of it, and apply our sequential test to decide if that component is zero or not. If the sequential test is designed well, it will be able to make an accurate and quick decision. If the test deems a component non-zero, we stop sampling and decide that the alternative hypothesis is true. However, if all probed components are deemed zero by the test, we decide that the null is true.

It can be shown that this procedure returns the correct decision with probability $\geq 1 - \varepsilon$ whenever the signal strength μ scales as $\sqrt{\max\{p \log \frac{n}{s}, \log \frac{1}{\varepsilon}\}}$, where recall that p parametrizes the speed of change of the signal. That is to say, in the worst case when the signal is constantly changing (i.e. $p = 1$), the performance of adaptive and non-adaptive procedures are the same, and there is nothing to be gained. However, the adaptive procedure does take advantage of situations when the signal only changes slowly. Remarkably, in the extreme case of static signals ($p = 0$), the detection boundary no longer depends on n!

We illustrate these points by a simple numerical experiment, similar to the one presented in the previous section. As mentioned above, the error of the adaptive sensing procedure should drop below the value ε when the signal strength is around $\mu_{\text{limit}} = \sqrt{4p \log \frac{n}{s}}$ (the constant 4 comes from the detailed computations in [10]). Therefore we run both adaptive and non-adaptive tests in situations when the signal strength is $t \cdot \mu_{\text{limit}}$ with different values of t (around 1).

We run both procedures 100 times for every setting of t, and plot the average errors along with error bars in Figure 3. The error bars are constructed the same way as in the previous section. Our choice for the remaining problem parameters are $n = 2^{15}, s = 2^4$ as before, but now we consider the small sample regime $m = \frac{n}{s} \log_2(\frac{2}{\varepsilon})$ (the details behind the exact choice of m can be found in [10]). We run two different simulations, first when the speed of change p is equal to 0.2 and second when it is equal to 0.5.

As expected, the adaptive sensing test outperforms the non-adaptive one. The margin by which the adaptive sensing test is superior is greater as p decreases, which is what the theory suggests. Note that the error probability of the tests never quite reach the value zero, since with this choice of m, the probability of being able to sample an active component is not overwhelming.

Final remarks

In this paper we hope to have given the reader an idea of the potential of adaptive sensing, as well as the methodologies required for the development of sound algorithms. As mentioned earlier, the signal and measurement models can be significantly extended, going beyond Gaussian noise and purely sparse signals. Furthermore, instead of considering coordinate-wise measurements one can also consider linear ensembles — this is known as compressive sensing. Although technically more involved, similar results to the ones described here can be obtained in that setting as well [9] (although some open questions remain).

A related problem to the detection/estimation of means is the detection of correlations, where the aim is to determine whether there are correlated components within a high-dimensional vector [6]. In that scenario coordinate-wise measurements are not informative, and one always needs to sample multiple components simultaneously to determine if they are correlated. In that setting adaptive sensing can be advantageous when there is significant structure to the set of correlated components. However, it is an open question if in unstructured cases there is still a significant advantage to using adaptive sensing techniques.

References

1 Louigi Addario-Berry, Nicolas Broutin, Luc Devroye and Gabor Lugosi, On combinatorial testing problems, *The Annals of Statistics* 38(5) (2010), 3063–3092.

2 Maria-Florina Balcan, Alina Beygelzimer and John Langford, Agnostic active learning, *Journal of Computer and System Sciences* 75(1) (2009), 78–89.

3 S. Bubeck and N. Cesa-Bianchi, *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems*. Foundations and Trends in Machine Learning, Now Publishers, 2012.

4 Raied Caromi, Yan Xin and Lifeng Lai, Fast multiband spectrum scanning for cognitive radio systems, *IEEE Transactions on Communications* 61(1) (2013), 63–75.

5 Rui M. Castro, Adaptive sensing performance lower bounds for sparse signal estimation and testing, *Bernoulli* 20(4) (2014), 2217–2246.

6 Rui M. Castro, Gabor Lugosi and Pierre-Andre Savalle, Detection of correlations with adaptive sensing, *IEEE Transactions on Information Theory* 60(12) (2014), 7913–7927.

7 Rui M. Castro and Robert D. Nowak, Minimax bounds for active learning, *IEEE Transactions on Information Theory* 54(5) (2008), 2339–2353.

8 Rui M. Castro and Ervin Tánczos, Adaptive sensing for estimation of structured sparse signals, *IEEE Transactions on Information Theory* 61(4) (2015), 2060–2080.

9 Rui M. Castro and Ervin Tánczos, Adaptive compressed sensing for estimation of structured sparse sets, *IEEE Transactions on Information Theory* 63(3) (2017), 1535–1554.

10 Rui M. Castro and Ervin Tánczos, Are there needles in a moving haystack? Adaptive sensing for detection of dynamically evolving signals, arXiv:1702.07899, to appear in *Bernoulli*, 2017.

11 David Donoho and Jiashun Jin, Higher criticism for detecting sparse heterogeneous mixtures, *Annals of Statistics* 32(3) (2004), 962–994.

12 Simon Foucart and Holger Rauhut, *A Mathematical Introduction to Compressive Sensing,* Birkhäuser, 2013.

13 Aurlien Garivier and Emilie Kaufmann, Optimal best arm identification with fixed confidence, in Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, eds., *29th Annual Conference on Learning Theory, 23–26 June 2016*, Proceedings of Machine Learning Research, Vol. 49, Columbia University, pp. 998–1027.

14 Robert Gwadera, Mikhail J. Atallah and Wojciech Szpankowski, Reliable detection of episodes in event sequences, *Knowledge and Information Systems* 7(4) (2005), 415–437.

15 S. Hanneke, *Theory of Disagreement-Based Active Learning*, Foundations and Trends(r) in Machine Learning Series, Now Publishers, 2014.

16 Jarvis Haupt, Rui M. Castro and Robert Nowak, Distilled sensing: Adaptive sampling for sparse detection and estimation. *IEEE Transactions on Information Theory* 57(9) (2011), 6222–6235.

17 R. King, J. Rowland, S.G. Olivier, M. Young, W. Aubrey, E. Byrne, M. Liaka, M. Markham, P. Pir, L.N. Soldatova, A. Sparkes, K.E. Whelan and A. Clare. The automation of science, *Science* 324 (2009), 85–88.

18 Husheng Li, Restless watchdog: Selective quickest spectrum sensing in multichannel cognitive radio systems, *EURASIP Journal on Advances in Signal Processing* 2009(1), 417457.

19 Wuqiong Luo and Wee Peng Tay, Finding an infection source under the sis model, in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 2930–2934.

20 Matthew L Malloy and Robert D Nowak, Sequential testing for sparse recovery, *IEEE Transactions on Information Theory* 60(12) (2014), 7862–7873.

21 Vir V. Phoha, *Internet Security Dictionary,* Springer Science & Business Media, 2007.

22 Devavrat Shah and Tauhid Zaman, Rumors in a network: who's the culprit? *IEEE Transactions on Information Theory* 57(8) (2011.), 5163–5181

23 David R. Thompson, Sarah Burke-Spolaor, Adam T Deller et al., Real-time adaptive event detection in astronomical data streams, *IEEE Intelligent Systems* 29(1) (2014), 48–55.

24 Abraham Wald. Sequential tests of statistical hypotheses, *The Annals of Mathematical Statistics* 16(2) (1945), 117–186.

25 Heng Wang, Minh Tang, Yu-Seop Park and Carey E. Priebe, Locality statistics for anomaly detection in time series of graphs, *IEEE Transactions on Signal Processing* 62(3) (2014), 703–717.

26 Kai Zhu and Lei Ying, Information source detection in the sir model: A sample path based approach, in *Information Theory and Applications Workshop* (ITA), 2013, pp. 1–9.

Wouter Koolen
Machine Learning Group
Centrum Wiskunde & Informatica, Amsterdam
wmkoolen@cwi.nl

Sub-Gaussians in game-theoretic probability

In their elegant *Game-Theoretic Probability* framework, Shafer and Vovk interpret probabilistic assumptions as the availability of certain elementary bets, and derive probabilistic consequences by strategically combining these bets. In this article Veni laureate Wouter Koolen of the Machine Learning Group at CWI takes this framework for a spin by deriving (game-theoretic) deviation inequalities for sub-Gaussian random variables.

The purpose of this article is threefold. First, I will derive deviation inequalities for sub-Gaussian random variables. Such statements find application in statistics and machine learning, for example in hypothesis tests, confidence intervals and optional stopping. So if you have not seen sub-Gaussian variables or deviation inequalities (or both) before, this will be useful. Second, these results will illustrate how one can systematically exploit the (weak) assumption that the distribution belongs to a given set. And finally, the unified way in which the results are derived illustrates the power and intuitiveness of the game-theoretic probability framework.

The article is structured as follows. We will first review the sub-Gaussianity assumption, and investigate its game-theoretic interpretation as a collection of available bets. Subsequently, assuming that S is sub-Gaussian, we will construct betting strategies that lead to upper bounds on $\mathbb{P}\{S \geq c\}$, $\mathbb{P}\{S^2 \geq c\}$, $\mathbb{E}[e^{\lambda S}]$, $\mathbb{E}[e^{\lambda S^2}]$ as well as on $\mathbb{P}\{\sum_{i=1}^K S_i^2 \geq c\}$ for independent sub-Gaussian S_i. Each of these bounds expresses the intuition that S cannot be extreme with high probability.

Setup

The goal is to showcase the game-theoretic probability framework and techniques, and the associated way of thinking in terms of intuitive bets. Moreover, as we will see the constructions will come with natural certificates of tightness.

A minimal assumption

In this article we will not commit to a single distribution, but instead work with a (non-parametric) class of distributions. Here, the assumption that we are willing to make is that of sub-Gaussianity. Let's review the definition.

Definition. A variable S is *t-sub-Gaussian* if for each $\eta \in \mathbb{R}$,

$$\mathbb{E}[e^{\eta S}] \leq e^{t\eta^2/2}. \tag{1}$$

Sub-Gaussian random variables are ubiquitous. They are often used as models for noise (sub-Gaussianity implies mean zero). The centred Gaussian distribution with variance t satisfies (1) with equality, hence the name. Moreover, by Hoeffding's Inequality, any zero-mean random variable with bounded support $[a,b]$ is $(b-a)^2/4$-sub-Gaussian. For independent t_i-sub-Gaussian S_i, the sum $\sum_i S_i$ is $\sum_i t_i$-sub-Gaussian. For any $\alpha \in \mathbb{R}$ the scaling αS is $\alpha^2 t$-sub-Gaussian. The canonical t-sub-Gaussian random variable in applications is $S = \sum_{i=1}^t (X_i - \mu)$ where the $X_i - \mu$ are independent 1-sub-Gaussian.

Next, we will review the game-theoretic-probability interpretation of our assumption (1).

Game-theoretic probability

Following [3], we interpret the sub-Gaussian condition (1) as a collection of bets that are offered regarding S. Namely, for each $\eta \in \mathbb{R}$ we can buy any positive number of η-tickets that cost $e^{t\eta^2/2}$ and pay out $e^{\eta S}$. In other words, each unit capital invested in η-tickets yields $e^{\eta S - t\eta^2/2}$. A *strategy* for the learner is a *portfolio*, specified by a positive measure $p(\eta) \geq 0$, indicating how much capital should be invested in η-tickets for each $\eta \in \mathbb{R}$. (Our strategies are relatively simple as we are considering just a single round. See [3] for general multi-round protocols. We will use the notation for densities throughout for simplicity, although we will find we need both continuous and discrete measures.) The cost of the portfolio $p(\eta)$ is hence

$$\int p(\eta)\, d\eta,$$

and for each possible outcome $S \in \mathbb{R}$, its payoff is

$$\int p(\eta)\, e^{\eta S - t\eta^2/2} d\eta.$$

We will use portfolios to price arbitrary variables.

Upper price

Our goal is to show that S cannot be extreme with high probability. Our approach will be to fix a function $Y(S)$, expressing how extreme we deem S to be. The mechanism is then to construct a portfolio of tickets such that we end up with payoff at least $Y(S)$ no matter the outcome S. Given that all bets are fair at best, it will be highly unlikely that the strategy pays off significantly more than its cost. Formally, we define the *upper price* of Y to be the minimum cost portfolio

$$\bar{\mathbb{E}}[Y] := \min_{p(\eta)\geq 0} \int p(\eta)\, d\eta \qquad (2a)$$

subject to the 'super-replication' constraint

$$\forall S \in \mathbb{R}: \ \int p(\eta)\, e^{\eta S - t\eta^2/2} d\eta \geq Y(S). \qquad (2b)$$

As the name suggests, the upper price bounds the expectation from above. To see why, fix any optimiser p^* of (2). Taking expectation of (2b) under any t-sub-Gaussian distribution on S, we find

$$\begin{aligned}\mathbb{E}[Y] &\leq \int p^*(\eta)\, \mathbb{E}\big[e^{\eta S - t\eta^2/2}\big] d\eta \\ &\leq \int p^*(\eta)\, d\eta = \bar{\mathbb{E}}[Y].\end{aligned}$$

Now how to approach the optimisation problem above?

Duality

As the objective and constraint in (2) are *linear* in the portfolio $p(\eta)$, this is an (infinite) linear program. Like finite linear programs, this problem has an associated *dual problem* where the role of variables and constrains are swapped. (Duality is a rich concept in mathematical optimisation, see for example [1]. A useful analogy is perhaps the simplest duality relation, namely that the *maximum* over a set is also the *minimum* number that is larger than each member.) In our case the dual problem asks for a positive measure $b(S)$ on outcomes S that maximises

$$\max_{b(S)\geq 0} \int b(S)\, Y(S)\, dS \qquad (3a)$$

subject to the 'fair ticket pricing' constraint

$$\forall \eta \in \mathbb{R}: \ \int e^{\eta S - \frac{t\eta^2}{2}} b(S)\, dS \leq 1. \qquad (3b)$$

We will be using duality to certify optimality. A pair $p^*(\eta) \geq 0$ and $b^*(S) \geq 0$ satisfying the constraints (2b) and (3b) for which the values (2a) and (3a) coincide simultaneously solves (2) and (3) to optimality. We will call such a pair a *saddle point*.

Applications in the univariate case

We now use the above duality relationship to compute the sub-Gaussian upper price $\bar{\mathbb{E}}[Y]$ of the five variables Y of interest from the introduction. Throughout we assume that S is t-sub-Gaussian.

One-sided tail

Fix a threshold $c \geq 0$, and let

$$Y := \mathbb{1}\{S \geq c\}.$$

We claim that the upper price is $\bar{\mathbb{E}}[Y] = e^{-\frac{c^2}{2t}}$, as witnessed by the following saddle point:

$$p^* = e^{-\frac{c^2}{2t}} \delta_{\eta = \frac{c}{t}} \quad \text{and} \quad b^* = e^{-\frac{c^2}{2t}} \delta_{S=c},$$

where we write $\delta_{S=c}$ for the Dirac point-mass at c. Primal feasibility (2b) follows from

$$\min_{S\geq c} e^{-\frac{c^2}{2t}} e^{\frac{c}{t}S - \frac{c^2}{2t}} = 1$$

and dual feasibility (3b) from

$$\max_{\eta} e^{-\frac{c^2}{2t}} e^{\eta c - \frac{\eta^2}{2} t} = 1.$$

(Exercise: what are upper price $\bar{\mathbb{E}}[Y]$ and saddle point for $c < 0$?)

Primal optimality tells us that $\mathbb{P}\{S \geq c\} \leq e^{-\frac{c^2}{2t}}$, while dual optimality tells us that we cannot prove a tighter bound without changing the assumptions or technique. For example, if we know that $S \sim \mathcal{N}(0,t)$ is Gaussian, we find $\mathbb{P}\{S \geq c\} = \Psi(-c/\sqrt{t})$.

Two-sided tail

Fix $c \geq 0$. Let's look at the two-sided threshold

$$Y := \mathbb{1}\{S^2 \geq c\}. \qquad (4)$$

It would be natural to conjecture that the upper price $\bar{\mathbb{E}}[Y]$ is just twice that of a single tail. But in actuality it is less, especially so for small c. To say what it is exactly, let v and z be the value and optimiser of

$$v = \max_{z} \cosh(z\sqrt{c})\, e^{-\frac{z^2}{2}t}.$$

We claim that the value is $\bar{\mathbb{E}}[Y] = \frac{1}{v}$, as witnessed by the saddle point

$$\begin{aligned} p^* &= \frac{\delta_{\eta=z} + \delta_{\eta=-z}}{2v} \ \text{and} \\ b^* &= \frac{\delta_{S=\sqrt{c}} + \delta_{S=-\sqrt{c}}}{2v}. \end{aligned} \qquad (5)$$

Let's first check dual feasibility,

$$\begin{aligned}\max_{\eta} \frac{e^{\eta\sqrt{c} - \frac{\eta^2}{2}t} + e^{-\eta\sqrt{c} - \frac{\eta^2}{2}t}}{2v} \\ = \frac{\max_\eta \cosh(\eta\sqrt{c})\, e^{-\frac{\eta^2}{2}t}}{v} = 1.\end{aligned}$$

And let's check primal feasibility,

$$\begin{aligned}\min_{S^2\geq c} \frac{e^{zS - \frac{z^2}{2}t} + e^{-zS - \frac{z^2}{2}t}}{2v} \\ = \frac{1}{v} \min_{S^2 \geq c} \cosh(zS)\, e^{-\frac{z^2}{2}t} \\ = \frac{1}{v} \cosh(z\sqrt{c})\, e^{-\frac{z^2}{2}t} = 1.\end{aligned}$$

For exact Gaussian $S \sim \mathcal{N}(0,t)$, we have $\mathbb{P}\{S^2 \geq c\} = 2\Psi(-\sqrt{c/t})$.

Moment Generating Function

In the previous two sections we quantified that S cannot be extreme by giving upper bounds on probabilities of its tail events. Another way of expressing that S cannot be extreme is to bound its moment generating function. (Tail bounds would then follow by Chernoff's method). Fix $\lambda \in \mathbb{R}$. Let's consider

$$Y := e^{\lambda S}.$$

For a centred Gaussian $S \sim \mathcal{N}(0,t)$ with variance t, we would find $\mathbb{E}[e^{\lambda S}] = e^{t\lambda^2/2}$. Here we show that $\bar{\mathbb{E}}[Y] = e^{t\lambda^2/2}$ for t-sub-Gaussian S as well. The following is a witnessing saddle point

$$p^* = e^{\frac{\lambda^2}{2}t} \delta_{\eta=\lambda} \quad \text{and} \quad b^* = e^{\frac{\lambda^2}{2}t} \delta_{S=t\lambda}.$$

Dual feasibility follows by

$$\max_{\eta} e^{\frac{\lambda^2}{2}t} e^{\eta t\lambda - \frac{\eta^2}{2}t} = 1,$$

while primal feasibility is established by

$$\forall S \in \mathbb{R}: \ e^{\frac{\lambda^2}{2}t} e^{\lambda S - \frac{\lambda^2}{2}t} = e^{\lambda S}.$$

We find that the upper moment-generating function $\bar{\mathbb{E}}[e^{\lambda S}]$ is exactly that of a Gaussian. Hence the generalisation to sub-Gaussian comes for free.

Moment generating function of square

Now it gets interesting. Fix $\lambda \in [0,1]$. Let's consider

$$Y := e^{\lambda \frac{S^2}{2t}}. \qquad (6)$$

In contrast to what happened before, here the supports of the components of the saddle point are *continuous measures*. We claim the value is

$$\bar{\mathbb{E}}[Y] = \frac{1}{\sqrt{1-\lambda}}$$

as witnessed by the saddle point

$$p^* = \frac{1}{\sqrt{1-\lambda}}\mathcal{N}\Big(0, \frac{\lambda}{t(1-\lambda)}\Big) \text{ and}$$
$$b^* = \mathcal{N}(0,t).$$

Let's check primal feasibility. For all S,

$$\int e^{\eta S - \frac{\eta^2}{2}t} p^*(\eta)\, d\eta = e^{\lambda \frac{S^2}{2t}}.$$

Now let's check dual feasibility. For all η,

$$\int e^{\eta S - \frac{\eta^2}{2}t} b^*(S)\, dS = 1.$$

Finally, the values indeed agree and are equal to

$$\bar{\mathbb{E}}[Y] = \int p^*(\eta)\, d\eta = \int e^{\lambda \frac{S^2}{2t}} b^*(S)\, dS = \frac{1}{\sqrt{1-\lambda}}.$$

Interestingly, if S is Gaussian then S^2/t has a χ^2 distribution, and hence the moment-generating function of $\frac{S^2}{2t}$ is equal to $(1-\lambda)^{-1/2}$. So we are not losing anything by generalising to sub-Gaussian.

Application in the multivariate case

We conclude the exposition by looking at the simplest multi-variate case. For here something very interesting happens. The setup will be as follows. We consider independent $S_1, \ldots, S_K$ where S_i is t_i-sub-Gaussian. The joint outcome $\boldsymbol{S} = (S_1, \ldots, S_K)$ will be revealed at once, so there is no sequentiality to the problem. Before it is revealed, we can engage in a collection of bets on the outcome. For every $\boldsymbol{\eta} \in \mathbb{R}^K$, we will be able to buy any number of $\boldsymbol{\eta}$-ticket, which each pay off $\prod_{i=1}^K e^{\eta_i S_i}$ and cost $\prod_{i=1}^K e^{\frac{\eta_i^2}{2} t_i}$. So now a strategy for the learner is a positive measure $p(\boldsymbol{\eta})$ on $\mathbb{R}^K$. We will be interested in the statistic

$$Z := \sum_{i=1}^K \frac{S_i^2}{2t_i}.$$

This statistic arises for example as the maximum log-likelihood value when comparing arbitrary mean models with mean zero models.

Products

The univariate price for $e^{\lambda \frac{S^2}{2t}}$ developed below (6) immediately gives us a price for the product $\prod_{i=1}^K e^{\lambda \frac{S_i^2}{2t_i}} = e^{\lambda Z}$, namely

$$\bar{\mathbb{E}}[e^{\lambda Z}] = (1-\lambda)^{-K/2}.$$

Self-normalised sums of squares

We finally consider thresholding Z in the form of

$$Y := \mathbb{1}\{Z \geq c\}.$$

The upper price $\bar{\mathbb{E}}[Y]$ and witnessing strategies will need to generalise those below (4), which cover the case $K = 1$. This indeed happens, but in a curious way. Namely, the general pattern is to have $p(\boldsymbol{\eta})$ and $b(\boldsymbol{S})$ mix over certain ellipses. In the special case $K = 1$ we indeed recover the mixtures over 2 symmetrically placed points that we found in (5). To express the result, let v and d be the value and optimiser of

$$v = \max_{d \geq 0} e^{-d} \mathbb{E}_q\big[e^{2\sqrt{cd}q}\big]$$

where $q \in [-1, 1]$ has density

$$\frac{\Gamma\big(\frac{K}{2}\big)}{\sqrt{\pi}\,\Gamma\big(\frac{K-1}{2}\big)}(1-q^2)^{\frac{K-3}{2}}, \tag{7}$$

which we may recognise as the marginal density of a point drawn uniformly from the unit sphere $\mathcal{S}^K$ in $\mathbb{R}^K$. The final claim is that the upper price is $\bar{\mathbb{E}}[Y] = \frac{1}{v}$, as witnessed by the saddle point

$$p^* = \frac{1}{v}\mathcal{L}\Big(\boldsymbol{\eta}\colon \eta_i = \sqrt{2dt_i^{-1}}\, Y_i \text{ where } \boldsymbol{Y} \overset{\text{unif}}{\sim} \mathcal{S}^K\Big),$$
$$b^* = \frac{1}{v}\mathcal{L}\Big(\boldsymbol{S}\colon S_i = \sqrt{2ct_i}\, X_i \text{ where } \boldsymbol{X} \overset{\text{unif}}{\sim} \mathcal{S}^K\Big),$$

where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are uniformly distributed on the unit sphere. Here $\mathcal{L}(\cdot)$ denotes the law of the sampling procedure specified in the argument.

First, let's check dual feasibility. For all $\boldsymbol{\eta}$, abbreviating $z = \frac{1}{2}\sum_i \eta_i^2 t_i$ and $q = X_1$ (which has the density given in (7) above),

$$\frac{1}{v}\mathbb{E}_{\boldsymbol{S}}\Big[e^{\sum_i\left(\eta_i S_i - \frac{\eta_i^2}{2}t_i\right)}\Big] = \frac{1}{v}\mathbb{E}_{\boldsymbol{X}}\big[e^{\sqrt{2c}\sum_i \eta_i \sqrt{t_i} X_i - z}\big] = \frac{1}{v}\mathbb{E}_{\boldsymbol{X}}\big[e^{2\sqrt{cz}X_1 - z}\big] = \frac{1}{v}e^{-z}\mathbb{E}_q\big[e^{2\sqrt{cz}q}\big] \leq 1.$$

Okay, good. Now primal feasibility. We have

$$\frac{1}{v}\mathbb{E}_{\boldsymbol{\eta}} e^{\sum_i\left(\eta_i S_i - \frac{\eta_i^2}{2}t_i\right)} = \frac{1}{v}\mathbb{E}_{\boldsymbol{Y}}\Big[e^{2\sqrt{d}\sum_i \frac{S_i}{\sqrt{2t_i}} Y_i - d}\Big] = \frac{1}{v}\mathbb{E}_{\boldsymbol{Y}}\Big[e^{2\sqrt{d}\sqrt{\sum_i \frac{S_i^2}{2t_i}}\, Y_1 - d}\Big] \geq \frac{1}{v}\mathbb{E}_q\big[e^{2\sqrt{cd}q - d}\big] = 1.$$

In both cases the crucial step is to use rotational symmetry: for $\boldsymbol{c} \in \mathbb{R}^K$, the inner product $\sum_i c_i X_i$ has the same distribution as $\|c\| X_1$. Finally, note that

$$\mathbb{E}_q\big[e^{2\sqrt{s}q}\big] = {}_0F_1\Big(\frac{K}{2}, s\Big).$$

is the confluent hypergeometric limit function, for which computer support is readily available. For example, Mathematica calls it `Hypergeometric0F1`, Matlab calls it `hypergeom` and Octave has `gsl_sf_hyperg_0F1` in package `gsl`.

We found $\bar{\mathbb{E}}[Y] = \frac{1}{v}$, and hence $\mathbb{P}\{Z \geq c\} \leq \frac{1}{v}$. We can also reason backwards and find the threshold c corresponding to a given confidence $\frac{1}{v} = \delta$. We obtain

$$c^* = \min\Big\{c \,\Big|\, \max_{d \geq 0} e^{-d}\mathbb{E}_q\big[e^{2\sqrt{cd}q}\big] \geq \frac{1}{\delta}\Big\} = \inf_{s \geq 0} \frac{s}{\ln_+\big(\delta \mathbb{E}_q\big[e^{2\sqrt{s}q}\big]\big)},$$

which can be implemented numerically using a binary search for the zero of the derivative.

Conclusion

We illustrated the power of the game-theoretic probability framework by deriving in a uniform fashion a series of deviation inequalities for sub-Gaussian random variables. We covered just the tip of a giant (and partially unexplored) iceberg. In more advanced sequential settings, taking bets and observing outcomes are interleaved, and more elaborate strategies beyond mixtures are possible and necessary, naturally leading to martingales. Analogues of the methods showcased here can be used to prove more advanced deviation inequalities that e.g. hold for arbitrary exponential families, and hold uniformly over time [2].

Acknowledgements

This article benefited from discussions with Emilie Kaufmann (Inria Lille), Aurélien Garivier (IMT Toulouse) and Peter Grünwald (CWI).

References

1 S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

2 E. Kaufmann, W.M. Koolen and A. Garivier, Sequential test for the lowest mean: From Thompson to Murphy sampling, arXiv: 1806.00973, 2018.

3 G. Shafer and V. Vovk, *Probability and Finance — It's Only a Game!*, Wiley, 2001.

Jim Portegies
Department of Mathematics and Computer Science
Eindhoven University of Technology
j.w.portegies@tue.nl

Ergo learning

How to build machines that learn and think like humans or animals? In this article Jim Portegies discusses the approach of Misha Gromov, the so-called 'ergo project'. Then he will highlight some closely related research in developmental robotics and artificial curiosity. Finally, Portegies will discuss what he thinks are next steps in the search for universal learning programs, and will make a case for the study of learning by imitating, or analysis by synthesis.

A few years ago, a friend told me about the online chat program CleverBot. The next few days I (embarrassingly) spent hours chatting with the bot, trying to prove in various ways that it wasn't actually a human I was chatting with, playing an interrogator at a Turing test. The conversations were absurd. CleverBot practically only reacted to the last sentence I said. I asked for its name several times and it gave me different answers. But this didn't rule out the possibility of chatting with a human (who was joking and couldn't hold the thread of a conversation). I was trying to make CleverBot say something that a human would never say. But I was doomed to fail, since CleverBot works with a system that reuses previous conversations with actual humans and so everything CleverBot said was basically said by a human at some point.

Surely, a Turing test would be set up differently: The human or machine behind the screen, or at the other end of the chat interface, would both try to convince the interrogator of their humanity. CleverBot is one of the best human-imitating chat programs around. But it doesn't pass such a Turing test by a long shot.

It means that Turing was too optimistic when he believed in 1950, that "...in about 50 years' time, it will be possible to programme computers, with a storage capacity of about 10^9, to make them play the imitation game so well that an average interrogator will not have more than 70% chance of making the right identification after five minutes of questioning" [24].

It intrigues me that, despite the many recent successes in machine learning and artificial intelligence, humans and animals still outperform machines on a large number of tasks. Humans are much better at generalizing from a small number of examples and carrying skills over from one task to another, and they are much more efficient doing so in terms of usage of energy, computational power and memory [13].

How to build machines that learn and think like humans or animals? How to design machines that pass the Turing Test? How to solve the grand problem of artificial intelligence, of designing artificial agents that interact optimally with real-world environments machines, given *limited resources* [19]? How to interpret machine-learning models [4, 11]?

It was another friend who told me about the approach of Misha Gromov to these questions. Until then, I had known Misha Gromov as a mathematician, for his work in geometry, and I did not know about his interest in artificial intelligence. I looked up the article on Gromov's website, and it somehow resonated with me. Over the last years, Gromov has posted and updated his articles several times [7, 8]. Recently, he bundled his thoughts in a book *Great Circle of Mysteries: Mathematics, the World, the Mind*, and at the time of writing I am waiting for the translation in English, to read Gromov's new account on what he calls the 'ergo project'.

The ergo project revolves around the conjecture that inside the human brain runs a simple, efficient, *universal learning algorithm* that applies indiscriminately to *any* incoming signal. That is, whether the incoming signal originates from the eyes,

the ears, or the sensorimotor system, it is processed in the same way. The algorithm finds *structure* in the incoming signals, finds *meaning* and *interpretation* and eventually leads to *understanding*.

A crucial part of the ergo project is to give *mathematical incarnations* to these last concepts. And whereas the idea is not new that inside the human brain runs a simple, efficient, universal learning algorithm, the vision of capturing a concept such as *meaning* in purely mathematical terms is rather original, and I think it is one of the aspects that sets the ergo project apart.

Another distinguishing factor of the ergo project is the big role of mathematics, and moreover, for mathematics yet to be developed. As such, it holds the promise to stimulate the development of a new mathematical field.

In this article I will discuss Gromov's ergo project in more detail. I will then highlight some closely related research in developmental robotics and artificial curiosity. Finally, I will discuss what I think are next steps in the search for universal learning programs, and in particular I will make a case for the study of *learning by imitating*, or *analysis by synthesis*.

Non-universal versus universal learning

Before I try to explain what universal learning is, let me first illustrate *non-universal* learning. Suppose we want to program a robot with a camera to navigate through a room. We could read the signals recorded by the robot's camera and immediately interpret them as light intensities on a grid. When we combine it with all our understanding of elementary optics and geometry of three-dimensional and projective space, it will be possible to hardcode a fairly successful navigation procedure. And this is currently done, of course, in many practical and toy robots.

In a theory on human learning called nativism, just like we know how to deal with the signals coming from the camera, the brain of an infant is assumed to be already preprogrammed to deal with the incoming signals.

The ergo philosophy is different. It is very similar to how Turing envisioned that a child's brain is a rather little mechanism, with lots of blank sheets: the ergo project is also surrounded by the expectation that in fact, very little preprogramming is present in the infant's brain. Without such preprogramming, the flow of signals entering a child's brain is much like a chaos of electrochemical sparks. Yet then, gradually and miraculously, an immensely powerful universal learning algorithm brain finds redundancies, patterns, structure in the chaos, starts to autonomously build interpretation, and finally starts understanding the incoming flow of signals.

Ego versus Ergo

Since this process of building understanding seems much stronger in early years of life, the expectation is that at least for adults, the ergo brain, that powerful universal learning algorithm, is dominated by other processes in the mind. According to Gromov, the mind roughly decomposes into two competing parts, the ego mind and the ergo brain, and this is certainly a helpful, and at times inspiring, metaphor.

The ego-mind is responsible for a person's primary needs, such as the needs for survival and reproduction. The ego-mind is at the surface of the brain, in our consciousness, and the part that we are most used to.

In the depths of our minds, as if hidden behind a wall, runs the ergo brain. Within this metaphor, we can explain the extraordinary capabilities of savants and of highly gifted mathematicians such as Ramanujan by cracks in the wall, through which the ergo brain shines through. And finally, children are all little Ramanujans, their task of finding structure in the chaos of electrochemical sparks harder and more abstract than the hardest mathematics.

The expectation is therefore that if we want to build universal learning algorithms, we should mimic how children learn. Children explore the world in an active, playful and curious manner. Their learning seems to be goal-free and independent of external rewards. They get bored of situations that they understand well, and stay away from situations which are too unpredictable.

To get an idea of the competition between ego and ergo, it is illustrative to look at the list of words Gromov associates with the ego mind,

intuitive, intelligent, rational, serious, objective, important, productive, efficient, successful, useful,

and those that belong to the ergo vocabulary,

interesting, meaningful, informative, funny, beautiful, curious, amusing, amazing, surprising, confusing, perplexing, predictable, nonsensical, boring.

With this division in ego and ergo comes of course a belief that in order to find universal learning algorithms one needs to follow an approach fitting the latter list of adjectives.

Let me now describe how we can go from metaphors closer to mathematics.

The ergo-learning conjecture

Gromov's ergo-learning conjecture states that (rephrased):

There exists a simple, *efficient*, universal *learning algorithm that finds structure*, *meaning and interpretation in any incoming stream of signals and finally converges to understanding*.

Universal means that no matter whether the signals come from visual, auditory or sensory input, or the signals are even generated internally, they are all processed using the same algorithm. Simple means logically simple, of low complexity. Efficient means that it can be realized within the human limits on computational, memory and energy resources.

What the words structure, meaning, interpretation and understanding mean is at this point unclear. So while working towards proving the conjecture, one needs to develop a mathematical theory of structure in signals. Similarly, one would need to develop mathematical concepts capturing the terms *understanding*, *interpretation*, *meaning* and *learning*.

The ultimate aim of the ergo project is to prove the conjecture by actually designing and implementing a universal learning algorithm. Such an ergo system would, on encountering *any* incoming flow of signals, start to interact with the flow and find meaning and understanding inside.

How plausible is ergo-learning conjecture?

This question can spawn a discussion that fits right into the nature-nurture debate. It is the discussion about how much preprogramming there is in an infant's brain, and how much understanding of the world is acquired through experience.

The main argument in favor of ergo learning is that evolution has not had

enough time to construct targeted learning algorithms for every possible signal and every possible task. Instead, an infant's brain is endowed with a simple, universal program that works for several signals. This is very similar to Turing's vision: he wrote that "Our hope is that there is so little in the child brain that something like it can be easily programmed."

As a reaction to psychological theories that were hard to test experimentally, scientists in the beginning of the twentieth century developed a new approach to psychology called behaviorism. It is far on the nurture end of the nature-nurture spectrum, and environmental factors, and in particular reinforcement by rewards and punishment, play a huge role [22].

Turing speculated that a learning machine could also be taught by using rewards and punishments and as such reflected the behaviorist point of view. Nonetheless, he himself indicated that other learning mechanisms would be necessary as well.

In 1957, Skinner published his behaviorist account on language processing [23] which prompted a famous critique by Chomsky [2]. Whereas in the behaviorist point of view, language is thought to be required solely through experience, the criticism by Chomsky was that children are not exposed to enough linguistic data to learn the features of the language, an observation he later called *Poverty of the stimulus*. Rather, key linguistic features should be innate, should be preprogrammed in the genetic structure. In particular, infants should have internalized a certain universal generative grammar.

There is currently no mathematical theory to support either side of the debate, but in this context I always think that it is astounding that the (naïve?) information content of a person's DNA is about 750 MB: it fits on a DVD. (This estimate is based on a total length of the human genome of approximately 3 billion base pairs, with each basepair accounting for two bits.) Many contemporary software packages will take significantly more space on the hard disk on your computer than that. It gives some indication of the limits on the complexity of the 'start-up' software of children.

Another strong hint at universality is that humans can learn language even when they are deaf or deaf-blind. This means that language can be learned independently of the signal carrying the linguistic information. Moreover, some humans have an ability to learn complicated structures, such as how to play chess, by pure observation. Many humans have an ability to learn mathematics. Since these are rather new, it is unimaginable that they are somehow encoded in the DNA.

There are also some claims and indicators of non-universality in human signal processing. The initial layers responsible for processing visual and auditory signals seem adapted to the type of signals they are processing. For instance, scientists such as Petitot claim that the connectivity and the geometrical arrangements in the first layers of the human visual system are crucial in processing the visual signals [17]. Moreover, the cochlea in the inner ear performs a type of Fourier transform of the incoming sound. Here, preprocessing of the signal happens even before the signal enters the neuronal system.

In contrast to this observation of non-universality, there is the observation that the brain shows a remarkable ability to adapt itself to new signals, an ability often referred to as 'brain plasticity'. For instance, there have been experiments with ferrets where the scientists connected the visual input to the auditory cortex. These ferrets could still 'see'. Moreover, the connectivities in the auditory cortex changed and formed a structure characteristic for the first layer(s) of the visual cortex [15]. This could be a reflection of ergo learning.

Ultimate aim

To me, the ultimate outcome of the ergo project are good mathematical definitions of *structure*, *meaning*, *interpretation* and *understanding*, and a 1 GB USB stick with an actual universal learning algorithm `Prog`. We should be able to install the universal learning algorithm in the 'brain' of an arbitrary robot, where the robot-brain is just a general-purpose computer with human-scale computation and memory capacity.

One robot may have a microphone, another may have a camera, a third may just have some pressure sensors. Some may produce sound, others images, some can move around and maybe others are completely stationary. We then let the robots interact with the environment, in particular, we expose them to sound recordings of text, let them flip through books. By running the algorithm `Prog`, the robots will start to find structure in the incoming signals, start to interact with it through its outgoing signals, and, within a few years, will converge to a form of understanding of language, for instance.

The design of a universal learning program is highly challenging, and some will believe that it is impossible. It is, in fact, even hard to envision at this stage what such a program may look like.

A robot learning to read

In Gromov's setup, 'understanding' is a combinatorial structure in itself, by which

Figure 1

we mean something like (but different from) a graph, a simplicial complex or an n-category. When the universal learning program is listening to the incoming flow of signals, the ergo-learning program will try to incorporate the structure that is diluted in the flow, in the combinatorial structure of the 'understanding'.

Suppose that the ergo-learning program `Prog` is connected to a large body of text, such as Wikipedia. The program starts by identifying textual units, such as words or word combinations that are persistent in the text. Importantly, we don't tell the program what a word is, or even a 'space'. It should perform the division into units purely based on statistical properties of the text.

Next, it would attach names or tags to some of these words and fragments. One can think of such tags being written on a line above the original string. One can iterate the annotation process and write $(l+1)$st order tags on top of the line with lth order tags. The number of tags should decrease with l.

The next step is the compression of the library, that is, the applications of several structure-preserving reductions, which results in a more elaborate combinatorial structure. This combinatorial structure reflects the grammar. It should be significantly smaller than the original body of text.

The resulting combinatorial structure will be a dynamically changing entity, which we will denote by $\mathcal{U}_t$. The program is iteratively applied to $\mathcal{U}_t$ to obtain the combinatorial structure at the next time point. Eventually, we expect this process to approximately converge to a fixed point. Then, we would say that the ergo brain 'understands' the language.

Generalizing from the above, the whole process of understanding would consist of three components:

- A combinatorial structure $\mathcal{U}$ in the mind, brain or program that 'understands';
- an ergo-system that implements $\mathcal{U}$, i.e. takes $\mathcal{U}$ as an input and uses it to process and interpret incoming (and internally generated) signals;
- the result of applying the implementation of $\mathcal{U}$ to incoming flows of signals.

Developing reasonable understanding of a language takes a long time, at least several years for humans. A similar timeframe holds for the understanding of a mathematical theory. That is why Gromov expects that the time-complexity of a learning process is log-linear in the size of the language or theory. On the other hand, the applications of the understanding to a flow of signals is extremely fast, and therefore expected to be logarithmic in the size of the signal.

Even though some will be skeptical about the feasibility of the ergo project, closely related research has been going on for decades, albeit under different names such as *artificial curiosity* [21], *developmental robotics* [3,12] and the *Bayesian brain hypothesis* [5,6]. (In fact, Jürgen Schmidhuber's survey article on artificial curiosity has the surprising title 'Formal theory of creativity, fun, and intrinsic motivation (1990–2010)'.)

Going back in time even further, Turing's original proposal of teaching a machine by rewards and punishments was made operational in a technique called reinforcement learning. Reinforcement learning in itself is not quite universal, but it can play an important role in more universal learning algorithms, so we review it first.

Reinforcement learning

Suppose we want to let the robot perform a certain task, such as play a game of soccer or chess. We could try to hardcode rules of play, program the robot to perform optimally, but this is a very non-universal approach. Instead, we let the robot figure out by itself how to play, except we, as external experts, do 'grade' the performance of the robot, and make our grading accessible to the robot, through a part of its input signal.

We can program the robot in such away that this part of the input signal gets the interpretation of a reward, which the robot tries to maximize (often on average). This approach goes by the name of *reinforcement learning*, and is a reflection of the behaviorist principle of learning through rewards and punishments.

Reinforcement learning has grown into an extremely powerful method to teach tasks and games to machines and robots [18].

Reinforcement learning has both universal and non-universal aspects. It is universal in that the same algorithm can be used on a large variety of tasks. However, reinforcement learning is predominantly used with outside-programmed reward signals that indicate performance on a certain task. In that sense it is very (external) goal-oriented, task-specific and therefore not really universal.

Artificial curiosity

Reinforcement learning is analogous to teaching an animal a trick by giving it food in return. In such a case, the reward signal is coupled to a primary (or secondary...) drive of the animal.

Around the fifties, psychologists started to realize that it is hard to explain the behavior of children, humans and animals by merely goal-oriented behavior. Experiments showed that animals had a craving for novel experiences and were behaving in exploratory ways even when their primary needs seemed to be satisfied, when exploratory behavior actually led to punishments, or when there really seemed to be no reward for an activity at all [1,25].

If children and animals are not solely motivated by external rewards, then how do they learn? And how could we mimic such behavior in robots? In particular, how can we instill exploratory behavior in robots, a craving for novel, surprising, funny experiences, and a certain focus on situations that are interesting because they are not too predictable nor too unpredictable? How can we artificially generate curiosity?

Interestingly, very similar suggested answers to these questions come from many different directions. The core of these suggestions is as follows: Endow the robot with a predictor of its own sensory input and choose its actions based on the accuracy of the predictions. The robot can measure this accuracy by itself, without the need for an external grader.

The above principle comes in many different flavors, as it depends on a particular system of prediction used and the particular quantities that are optimized and how the optimization procedure takes place.

A very interesting application was constructed by Oudeyer, Kaplan and Hafner [16]. Their robot seemed to go through various stages of development. If it would start to 'understand' a certain toy or situation, it would move on to the next, where there still was something new to learn.

Prediction and generation

The predictor is a key component of the above setup. It *generates* a signal that

Figure 2

can be compared to the true input. I first want to describe the simplest such predictor, which already contains the key idea of much more sophisticated versions.

We think of showing the robot a sequence of uniform grayscale images, according to the following procedure. To begin with, we choose a random 'original' picture according to a probability distribution that we call the *prior probability distribution*. There are just two choices for the original picture: it is either completely white or completely black. We call this picture Or and we keep it hidden from the robot. It stays fixed throughout the rest of the procedure.

We then show the robot a sequence of 'noisy' images Im_1, Im_2, ..., in which the image is colored in a random gray scale in $\{0,1,...,10\}$, where 0 corresponds to black and 10 to white. This means for instance (just to give a concrete example) that if the original image is black, the probability that the grayscale of image j equals m is proportional to

$$\mathbb{P}[\mathsf{Im}_j = m \mid \mathsf{Or} = B] \sim \left(\frac{9}{10}\right)^m$$

and if the original image is white, the probability that the grayscale of image j equals m is proportional to

$$\mathbb{P}[\mathsf{Im}_j = m \mid \mathsf{Or} = W] \sim \left(\frac{9}{10}\right)^{10-m}.$$

The noise for image i is independent of the noise for image j if $i \neq j$.

The first ten images may look like shown in Figure 2. In inference, it is the task of the robot to infer what was the original picture, based on such a sequence of pictures. In prediction, it is the task of the robot to predict how the sequence continues.

The random picture Or is often called a latent variable. It 'explains' the sequence of images produced.

We first assume that the robot has full knowledge about the above procedure, including the various probabilities, so it can use this knowledge to do inference and prediction. Let us first see how inference works.

After the robot sees one image with gray scale m, it can calculate the probability that the original image was the black image B using Bayes' formula

$$\mathbb{P}[\mathsf{Or} = B \mid \mathsf{Im}_1 = m] = \frac{\mathbb{P}[\mathsf{Im}_1 = m \mid \mathsf{Or} = B]\,\mathbb{P}[\mathsf{Or} = B]}{\mathbb{P}[\mathsf{Im}_1 = m]}.$$

This conditional probability distribution on the original picture is called the *posterior probability distribution*.

After viewing more images, the robot can in the same way calculate the posterior probability that the original image was black, given observations of the images $\mathsf{Im}_1, \ldots, \mathsf{Im}_n$. The estimate the robot makes this way becomes more and more accurate (in fact exponentially fast). This is the essential idea of Bayesian inference.

Let us now look at Bayesian prediction. In this simple case, where the robot has full knowledge about the model generating the images, the robot uses that same generative model for its predictor: The robot's predictor generates a sequence of images exactly following the same procedure as above. After observing images, however, it will update its predictor: it uses the exact same generating process except for replacing the *prior* probability distribution for picking the original image by the *posterior* probability distribution. This way, after a few observations, the predicted signal of the robot will be much closer to the true signal.

This was a very simple example, but it contains the basis of Bayesian inference and prediction. Let us now make it a bit more complicated.

We are going to generate a sequence of images of arrangements of overlapping boxes. We first sample a random number N, which will be the total number of boxes in the picture, and for each of the boxes, we choose a random size. We keep both the number and the sizes fixed (and will sometimes refer to them as the fixed latent variables). To generate images, we vary the

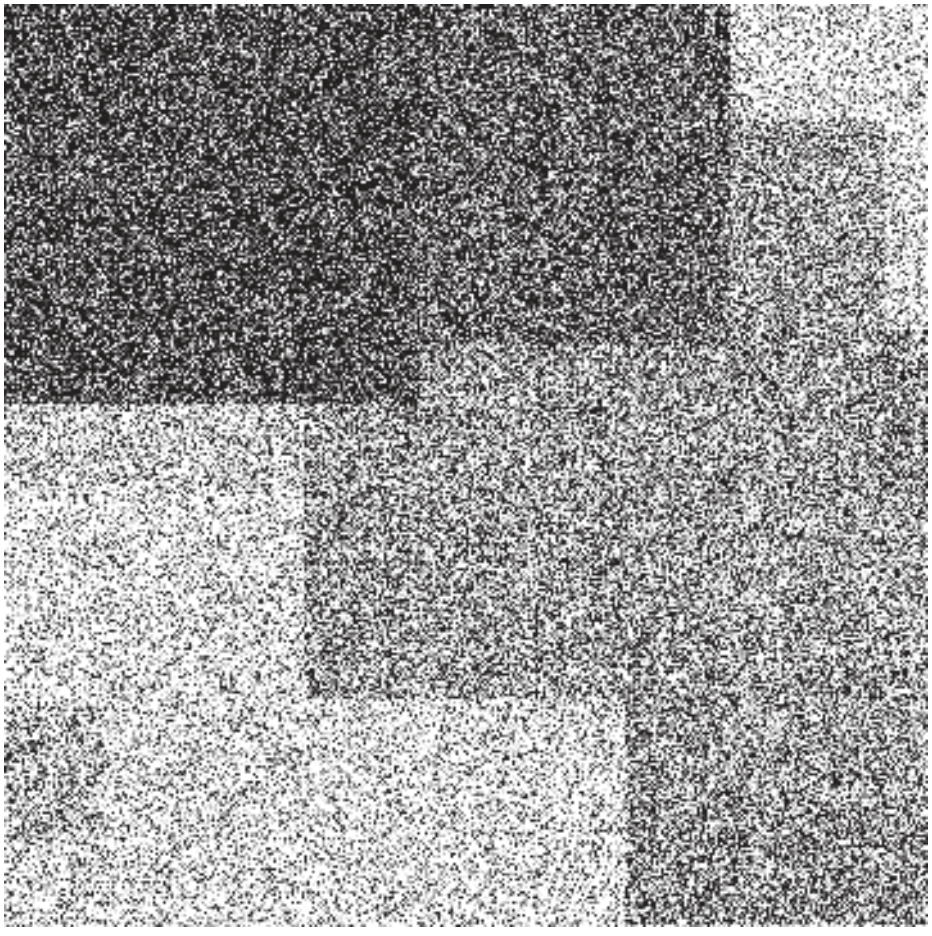


Figure 3

gray-scale, the xy-location, and an order parameter h of the boxes at random (and will refer to them as changing latent variables). A box is covering another box if the order parameter of the first box is larger than that of the second. Finally, we add Gaussian noise to every picture. A resulting picture may look like Figure 3, can you infer the number of boxes?

The various continuous variables present in the problem make it very hard to do Bayesian inference directly. For general distributions, it is impossible for the robot to find a good representation of the posterior distribution. There are several ways to deal with this problem, one of which is by sampling, and the other is by a technique called *variational inference*, where one approximates the true posterior distribution by an element of a simpler, parametrized family of distributions.

If the robot has full knowledge of the model, it can by observing the sequence of images obtain ever-more accurate estimates of the fixed latent variables: the number of boxes in the images and their sizes. For every single image, the robot can estimate the changing latent variables, namely the position of the boxes and their grayscale. In general, this procedure works quite well.

Analysis by synthesis

The situation changes drastically when the generating model is not known to the robot. Then we do not even know which variables to infer. Is there a way in which the robot may still find out the generative process?

My expectation is that this is possible if the robot is able to accurately approximate the signal (or rather its statistical properties), with a generative model that is as simple as possible. In that case, in the above example, the robot will in fact have encoded a certain number of boxes, a concept of size of the boxes, a concept of location and a concept of order. This way, the robot would have learned by constructing the signal itself, by applying analysis by synthesis.

The heuristic applied here is also known as Occam's razor, to select the simplest model explaining the observations. In the building where I work, there is restricted elevator access in the weekend. As my office is on the seventh floor, I soon discovered that I cannot go up to my office using the elevator without swiping my access card. Leaving the office, however, was never a problem. I stored this rule (and this was a golden truth to me) as "in the weekend you can go down, but you can't go up". Until, of course, a visitor who took the stairs got stuck on the third floor, and it turned out I cannot go down from my office to get him. I came up with a new rule (you can always go to floor 0 and 1, but not to any other floor) and am again utterly convinced of its truth, even though I never tested whether you can go from floor 4 to floor 6.

There are several layers to this story, but the main message is that humans have an uncanny ability to generalize from a very limited number of examples, favoring simpler explanations. The models that we build for ourselves are often wrong, but it doesn't matter much until we encounter new evidence, and we easily build a new simple model. It is a real challenge to get robots to do something similar.

One way is by letting the robot perform *inductive inference*. In a way, inductive inference is Bayesian inference taken to an extreme: because the robot does not know what is the true generative model, it adds the model itself as a (fixed) latent variable! In this case, the robot's generator first selects at random *any* sequence-producing model, and then samples from this model as before. The robot implements Occam's razor by taking the complexity of the models into account in the prior probability distribution: 'simpler' models are selected with higher probability.

There are, of course, a few problems with this approach. The space of all sequence-producing models is too large to handle effectively. In addition, one needs to have a workable concept of complexity of models, and although in this generality there are beautiful mathematical definitions such as Kolmogorov complexity, the fact that the Kolmogorov complexity cannot be computed is problematic.

Nonetheless, Hutter introduced a rational reinforcement learning agent, called AIXI [9], completely based on inductive inference. The AIXI algorithm is more like an abstract, mathematical object, since the model is uncomputable. Hutter also introduced a computable procedure, called AIXI(t,l), but it suffers from very large computation times. Schmidhuber developed the Gödel machine partly to deal with this issue of computation time [20]. The Gödel machine can rewrite its own software, after it has proven that such a rewrite is useful for rewards and speedup.

Although these learning algorithms are universal, even for the Gödel machine the computation times seem so large that its role in an ergo-learning system is questionable.

Variational autoencoders

I still believe that analysis by synthesis is possible by following the heuristic of Occam's razor: if the robot does not know the generative model producing the data, it tries to search over a whole class of generative models that best approximates the observed signal, but keeping in account the complexity.

However, instead of considering about every possible generative model as in inductive inference, I think it will be necessary to restrict to a class of generative models for which it is easier to define and compute measures of complexity, which are easier to train and easier to analyze mathematically, but still are capable of encoding efficiently and accurately a wide range of signals.

One such restricted class of generative models are generative models implemented by deep neural networks, such as the Variational Autoencoders introduced by Kingma and Welling [10].

In a Variational Autoencoder, one completely artificially adds a (changing) latent variable Z to the system, often with values in $\mathbb{R}^n$. One then optimizes over a whole class of generative models which generate signals by sending the latent variable, together with a noise variable, through a deep neural network; different choices of parameters of the neural network give rise to different generative models. One adapts the parameters of the neural network so that the true signal is approximated well, but it is not the only quantity that is optimized. At the same time, one uses a second deep neural network for variational inference: that is, one approximates the true posterior distribution by the family of distributions that one gets by varying the parameters of the second network. (This most basic setup of a Variational Autoencoder takes into account the complexity of Z — through the principle of minimum description length —, but not of the generative model itself. One could,

instead, incorporate the complexity of the generative model by also adding the parameters of the neural network as fixed latent variables in a Bayesian setup. This would be closer to inductive inference. One can even iterate the procedure, and get models similar to the hierarchical models Friston proposes in his variational-inference approach to the Bayesian brain hypothesis [5].)

Together with Vlado Menkovski and Luis Pérez, we are currently investigating the use of Variational Autoencoders for analysis by synthesis. For instance, if in the example with the pictures with boxes, we fix the number of boxes at 1, we expect that from the latent variable Z we can read off the x-coordinate, the y-coordinate and the grayscale of the box.

The x-coordinate will almost certainly not correspond to the first component of Z, but (in a slightly idealized case) we can find a (linear?) change of coordinates Φ from $\mathbb{R}^n$ to $\mathbb{R}^n$ so that the first, second and third component of $\Phi(Z)$ correspond to the x-coordinate, y-coordinate and the gray scale of the box, respectively.

This way, we are naturally led to questions about when two generative models are equivalent, about when there exists a translation from one generative model to the other and back. This opens the door to more mathematical definitions of interpretations of models and structures diluted in signals.

Indeed, together with Rostislav Matveev we looked at a definition of structure in signals for a certain type of translation between signals relevant for information processing [14]. If a signal consists of multiple components, the concept of structure we define naturally includes the Shannon mutual information between the components. However, it also covers finer dependency structure relevant for information processing.

Conclusion

It is clear that the ergo project is in its infancy, but there seem to be a few directions in which we can try to make small steps forward, such as in the search for relevant definitions of structure in signals, and in the development of theory and understanding of the process of analysis by synthesis, for instance in the context of deep generative models.

My personal belief is that in some decades from now, the ergo project and related activities will have lead to a new branch of mathematics, in which words such as *structure*, *meaning*, *interpretation* and *understanding* are mathematical concepts, and the only reason that we won't be able to construct a 1 GB USB stick with a universal learning program `Prog` that can be used to let a robot find structure and eventually understand a wide range of flows, will be that 1 GB USB sticks are no longer to be found.

References

1 Daniel E. Berlyne, Novelty and curiosity as determinants of exploratory behaviour, *British Journal of Psychology* 41(1-2) (1950), 68–80.

2 Noam Chomsky, A review of B. F. Skinner's Verbal Behavior. *Language* 35(1) (1959), 26–58.

3 Angelo Cangelosi, Matthew Schlesinger, and Linda B. Smith, *Developmental Robotics: From Babies to Robots*, MIT Press, 2015.

4 Finale Doshi-Velez and Been Kim, A roadmap for a rigorous science of interpretability, arXiv:1702.08608, 2017.

5 Karl Friston, The free-energy principle: a unified brain theory?, *Nature Reviews Neuroscience* 11(2) (2010), 127–138.

6 Karl Friston, The history of the future of the bayesian brain, *NeuroImage* 62(2) (2012), 1230–1233.

7 Misha Gromov, *Structures, Learning and Ergosystems*, Chapters 1–4, 6, www.ihes.fr/~gromov/PDF/ergobrain.pdf, 2011.

8 Misha Gromov, *Memorandum Ergo*, www.ihes.fr/~gromov/PDF/ergo-cut-copyOct29.pdf, 2015.

9 Marcus Hutter, *Universal artificial intelligence: Sequential decisions based on algorithmic probability*, Springer, 2004.

10 Diederik P. Kingma and Max Welling, Auto-encoding Variational Bayes, arXiv:1312.6114, 2013.

11 Zachary C. Lipton, The mythos of model interpretability, arXiv:1606.03490, 2016.

12 Max Lungarella, Giorgio Metta, Rolf Pfeifer and Giulio Sandini, Developmental robotics: a survey, *Connection science* 15(4) (2003), 151–190.

13 Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum and Samuel J. Gershman, Building machines that learn and think like people, *Behavioral and Brain Sciences* 40, 2017.

14 Rostislav Matveev and Jacobus W. Portegies, Tropical limits of probability spaces, part I: The intrinsic Kolmogorov–Sinai distance and the asymptotic equipartition property for configurations, arXiv:1704.00297, 2017.

15 Laurie von Melchner, Sarah L. Pallas and Mriganka Sur, Visual behaviour mediated by retinal projections directed to the auditory pathway, *Nature* 404(6780) (2000), 871.

16 Pierre-Yves Oudeyer, Frédéric Kaplan and Verena V. Hafner, Intrinsic motivation systems for autonomous mental development, *IEEE transactions on evolutionary computation* 11(2) (2007), 265–286.

17 Jean Petitot, The neurogeometry of pinwheels as a sub-Riemannian contact structure, *Journal of Physiology-Paris* 97(2-3) (2003), 265–309.

18 Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, Volume 1, MIT Press, 1998.

19 Jürgen Schmidhuber, Towards solving the grand problem of AI, *Soft Computing and Complex Systems*, 2003, pp. 77–97.

20 Jürgen Schmidhuber, Gödel machines: Fully self-referential optimal universal self-improvers, in *Artificial General Intelligence*, Springer, 2007, pp. 199–226.

21 Jürgen Schmidhuber, Formal theory of creativity, fun, and intrinsic motivation (1990–2010), *IEEE Transactions on Autonomous Mental Development* 2(3) (2010), 230–247.

22 Burrhus F. Skinner, *Science and Human Behavior*, Simon and Schuster, 1953.

23 Burrhus F. Skinner, *Verbal Behavior*, Appleton-Century-Crofts, 1957.

24 A.M. Turing, Computing machinery and intelligence, *Mind* 59(236) (1950), 433.

25 Robert W. White, Motivation reconsidered: The concept of competence, *Psychological review* 66(5) (1959), 297.

Martijn van Otterlo
Department of Cognitive Science and Artificial Intelligence
Tilburg University
m.vanotterlo@uvt.nl

Ethics and the value(s) of Artificial Intelligence

In this article Martijn van Otterlo provides an introduction to some current issues in the ethics of intelligent algorithms.

> "Solving the value-loading problem is a research challenge worthy of some of the next generation's best mathematical talent" [6, p. 229]

> "Supercharged with a larger cerebral cortex, faster learning, and a longer time horizon, is it possible that we solve complex problems in mathematics the same way that monkeys find optimal paths?" [29, p. 20152]

Artificial Intelligence (AI) [25] is booming. Although it has existed for more than six decades, recently it is literally everywhere. Phones have 'AI inside', computer games utilize 'AI opponents' and internet services let AI analyze posts and photos, personalize news feeds and find who or what you need. Each day news articles about AI appear, increasingly so since 2010 [11]. These advances have often profound consequences for our daily lives. For example, Google search fundamentally changed the way we obtain information [41] and Facebook's face recognition changed our personal privacy forever. Much of the current progress comes from the subfield of *machine learning* (ML) [19] and more specifically from a technique called *deep learning* (DL) [16] which has its roots in earlier *neural networks*. ML makes use of *data* to *inductively generate predictive models*. For example, based on a huge dataset of images, computers can nowadays be *trained* to recognize various items on pictures, and companies such as Facebook are heavily investing in such technology.

Anticipations of AI: good and bad

It is hard to predict where things are heading with all this progress in AI. Terry Sejnowski wrote in 2010 that *reinforcement learning* (RL) [46], a special kind of ML, had just beaten a 5*th dan* human professional in the board game *Go*, using similar techniques used to learn *backgammon* much earlier [35]. AI experts predicted that beating the human overall Go champion would take a decade at least when suddenly in 2016 the DL *AlphaGo* program [31] did just that after first beating humans at *Atari* games. But whereas AlphaGo learned from lots of human moves, its successor *AlphaZero* [32] returned to ideas that solved backgammon, and taught *itself* by just playing against incrementally better versions of itself and beat everyone, including AlphaGo, 100 against 0. Interestingly, the same ML method also dominates (human and machine) in *chess* after only 4 hours of training, from scratch, by itself, only 20 years after the heavily engineered IBM's *DeepBlue* algorithm beat the human champion Gary Kasparov.

Predicting developments is hard [14], but it is important to anticipate what can happen if AI achieves general *human-level intelligence*. Half of the experts in a 2013 panel predicted it to happen between 2040–2050, and most experts predicted *superintelligence* [6] would follow 30 years later, but 30 percent did not see this as positive for humanity [24]. Another study predicted human-level skills such as *translation* (by 2024) and *surgery* (by 2053) [17]. Negative consequences of AI, ranging from loss of jobs, to *malicious* use [48], and to superintelligence dominating humanity, have appeared in mainstream media too [11]. Until recently, criticism of AI (and especially ML) technology often came from the social sciences and legal scholars and was mainly about *data-related* concerns such as *privacy*, *surveillance* and *discrimination* [40]. More recently the broader implications of intelligent algorithms on society are studied [23, 44] and more importantly, by scholars from AI and ML itself. Recent reports explicitly take into account the *ethical* dimensions of AI for society [28, 34]. Recent books join: Tegmark's [37] (near and far AI future), Walsh's [45] on the status of AI, and Shanahan's overview [30] of the *singularity*.

Ethics of intelligent algorithms

Powerful ML algorithms can have profound influence in our digital society. Consider a case where a social network would approach users with a request to 'go nice' on a particular friend because it has statistically predicted that there is a higher than aver-

age probability that this friend has suicidal tendencies. This may sound *creepy* [38], but it is one of Facebook's recent plans: to predict potential suicides [50]. In a related effort, Google wants to detect depression [51]. Such predictions are technically interesting if they are possible, and possibly an opportunity to 'do good', but at the same time they *create* ethical issues: can a social network disclose or use such predictions without asking, and would that change people's behavior (of that friend, and towards that friend)? Even more ethically challenging, similar predictive capabilities can also be used to target insecure and troubled teenagers for *marketing purposes* [52]. Most people would not find that ethically correct, but some may see this as normal market practices. Other situations where powerful ML algorithms are used with ethical consequences are *search engines*, *personalized news feeds*, and *curation* on the internet. Examples include Facebook fighting terrorism [53], Google battling fake news [54] and Twitter's moderation [55]. All these technologies solve a problem: *information overload*. Without filtering and curating, humans could not handle the enormous amount of information. However, the ethical issue here is that these algorithms *select*, *hide* (e.g. the removal of the iconic 'napalm girl' photo due to Facebook's anti-nudity policy [56]), and *prioritize* sources for us. They basically decide what we get to see, and what not [41], which can be 'for good' again, but may equally well be considered *censorship*.

The *algorithmization* of society brings us many novel ethical issues. The study of societal consequences of AI beyond simple privacy and surveillance has become known as *ethics of algorithms* [23, 42, 44]. Sometimes *law* can be the answer, but so often it is far too slow to adapt [38] and we need to consider general ethical analysis. Algorithms basically transform *data* into *evidence*, which then is used to compute *actions*. Evidence can be inconclusive, inscrutable or misguided and this can cause many ethical consequences of actions, relating to *fairness*, *opacity*, *unjustified actions*, and *discrimination*. In the Facebook suicide case, evidence based on only Facebook data may be inconclusive and actions to act upon suspected suicidal tendencies may be unjustified. Overall, algorithms have an impact on *privacy* and can have *transformative effects* on *autonomy*. For example, the simple fact that Google selects our news may change how we think about particular subjects, and affect our autonomy to make decisions and trap us into *filter bubbles* [20]. It is useful to distinguish several core *classes* of algorithms I identified elsewhere [44]:

1. inference algorithms (*descriptive*),
2. learning algorithms (*predictive*),
3. optimization algorithms (*prescriptive*),
4. *superintelligence* [6, 30, 37].

A fifth class consists of *'physical'* algorithms such as *internet-of-things* and *robots*. Physicality will create another level of ethical problems [33], such as social backlash with surveillance robots [67] in public spaces or privacy issues with 'connected toys' [68]. Finding the right metaphor for how robots relate to non-physical *algorithms* may also help [27].

As said earlier, AI as a field is becoming aware of the issues itself. Novel ethics research centers arise [57], companies coordinate efforts [58], and scientists [59] and employees [60] speak out. In addition, education becomes aware [7, 47]. AI sub-communities have started to openly discuss the issues and identify *challenges* and *design principles*. The engineering community started the IEEE *Ethically Aligned Design* initiative [64], aimed at developing a *vision for prioritizing human well-being with autonomous and intelligent systems*, to capture concepts like *transparency* and *responsibility*, and to develop industry standards for *ethics*. The robotics community came with their own EPSRC *design principles* for robotics systems [66]. In January 2017, the AI community came up with their own *Asilomar principles* [65], a *code of ethics* which explicitly states 23 ethical principles for AI developers. The list includes general goals such as that AI should not target the creation of *undirected* intelligence, but instead should develop *beneficial* intelligence (principle 1). It also contains statements about *judicial* (principle 8) and *failure* (principle 7) transparency: an AI system should be able to *explain* its decisions.

Important *values* desirable in complex AI systems include *transparency* and *explainability*, to avoid *opacity* in decision making. Equally important are *responsibility*, *liability* and *accountability* [9] ("Who is to blame if an algorithm does harm?"). This should lead to *safe* and *trustworthy* AI systems. All these efforts should lead to professional *codes of ethics* for the development and employment of AI [5]. As I illustrated elsewhere [43], there is a strong parallel between these values (and intentions) and those behind *human* codes of ethics for various professions: they are used to openly and transparently communicate to the outside world what are the norms and values in a particular profession, and by doing that to earn trust and acceptance from outside. For increasingly intelligent AI, it is vital that not just the humans comply with the code, but the AIs too. For the latter, one literally obtains a *code* of ethics, embedded in the AI's program.

Ethical choices for an autonomous car in case of an upcoming accident

Towards building ethical AI systems

These *ethical* dimensions require AI designers to act *responsibly*. Some hope for a 'big red button' [49] to shut down 'rogue' AI systems, but that seems naive [4]. A better idea is to incorporate ethical thinking in the design of AI systems, possibly with the use of AI technology itself. For example, if a profiling algorithm is discriminatory,

one can modify it such that inherent biases are changed, or if autonomous cars can crash they should learn how to do it 'least harmfully'. Building ethical values into AI requires two things [6]:

1. a capacity to *acquire* ethical values, possibly from humans,
2. knowledge of *which human values* are important.

The field of AI is heavily invested lately in working on these two main issues. Some employ so-called *ethics bots* [10] to assist humans, while others focus on obtaining *human* ethical values through massive experiments [71]. In the following paragraphs, I briefly mention four current research directions.

AI and the good, the bad, science and the law

Fairness in machine learning

Many successful AI systems are based on ML, of which predictions can be biased by the data, parameters and application processes. A well-studied case concerns *recidivism risk score predictions* in the American judicial system [69], which sparked a lot of debate on what fairness actually is. In addition, it also showed that inherent biases in such decision making systems can have profound legal consequences. In relation to that, the notion of fairness has many connections to other concepts such as *diversity* and *discrimination*, which also makes it a highly interdisciplinary topic. Fairness is a *multi-objective problem* and intuitively, but also formally, algorithms that are fair on all possible accounts are impossible. Many different interpretations of *fairness* are being studied and much effort is being put into *fairness enhancing* techniques, to remove some of the unwanted biases in predictive algorithms [12]. A strong community has risen under the name of *fairness, accountability, and transparency in ML* (FAT) [62].

Explaining black box systems

On top of fairness and bias-related issues comes the fact that most powerful decision-making AI systems have become too complex to be *understandable* by humans, even though many of their decisions affect people's lives in various ways. Especially for deep learning systems there is a strong trade-off between *accuracy*, which contributes to the success of those models, and *interpretability*, which is hindered much by their complex, *black box* nature [8]. A current trend in ML is to investigate *interpretability* as a concept, and models that are (more) *interpretable* [70], sometimes capable of explaining the decisions of classifiers in human-understandable terms [26]. Much work is also being done on *extracting* (interpretable) knowledge *from* trained deep networks, for example by *distilling* [13], or other techniques that increase *transparency* and *explanation* [18]. Such approaches fit in a revived interest in so-called *explainable* AI [61].

Value-based AI

Much of the previous waves of attention to ethical issues dealt with a more limited set of aspects such as *privacy* and *surveillance* consequences of ML [40]. Currently, the focus is on the much broader notion of *value alignment* (VA): autonomous AI systems should be designed so that their goals and behaviors can be assured to align with human values throughout their operation. Obviously, VA is a *multi-objective optimization* problem too [39], and does view AI as an autonomous *agent* which takes *actions* and optimizes its behavior according to a *utility function*. Such settings are typical for the AI subfield of *reinforcement learning* (RL) [46]. RL is an ideal model for computational ethics [1], and many current VA issues studied in AI come from this area [2, 36]. Examples are

1. *scalable oversight* of ML systems by humans,
2. *mild optimization*, or 'not optimizing too hard',
3. *learning from humans*,
4. *safe exploration*.

RL is often used in combination with deep learning [21] (and because of that connects to all previously mentioned issues) but it can also be seen as an ideal suite of algorithms just because it is aimed at *value-based* optimization.

Ethical reasoning

The fourth direction towards ethical AI involves *reasoning*. As said, much of the success of current AI comes from *learning* approaches, but now (and in the past) these are being criticized [22] for their lack of explainability, and their incapability to insert and extract domain knowledge. Domain theories, models and formal logic, typically computer science tools, have been shown to be effective in the validation and verification of systems, and could be used to ensure their proper functioning [28]. It is only natural to consider logical models in the context of value alignment too, since they directly support *declarative*, *explainable* and *verifiable* reasoning of systems. AI has a rich tradition of such models, including those targeting *reasoning about ethical aspects* [3]. In recent work I introduced a novel approach combining formal logic, *decision-theoretic optimization*, and *supervised machine learning* for transparent (declarative), ethical reasoning in the face of uncertainty [43]. It is foreseeable that more such systems will follow to combine learning and reasoning about ethics in an explicit, and transparent way.

This text, but also the current state of the art, has only scratched the surface of what is needed to build truly ethical AI. The quotes at the beginning of this text show that this requires (mathematical!) advances to *build* such intelligent systems, but also to obtain the means to endow such systems with *our* (human) values.

References

1 D. Abel, J. MacGlashan and M.L. Littman, Reinforcement Learning as a Framework for Ethical Decision Making, in *AAAI Workshop: AI, Ethics, and Society*, 2016.

2 D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman and D. Mané, Concrete problems in AI safety, *CoRR*, abs/1606.06565, 2016.

3 M. Anderson and S.L. Anderson, Machine ethics: Creating an ethical intelligent agent, *AI Magazine* 28 (2007), 15–26.

4 T. Arnold and M. Scheutz, The 'big red button' is too late: an alternative model for the ethical evaluation of AI systems, *Ethics and Information Technology* 20(1) (2018), 59–69.

5 P. Boddington, *Towards a Code of Ethics for AI*, Springer, 2017.

6 N. Bostrom, *Superintelligence*, Oxford University Press, 2014.

7 E. Burton, J. Goldsmith, S. Koenig, B. Kuipers, N. Mattei and T. Walsh, Ethical considerations in AI courses, arXiv:1701.07769, 2017.

8 D. Castelvecchi, Can we open the black box of AI? *Nature News* 538(7623) (2016), 20.

9 N. Diakopoulos, Accountability in algorithmic decision making, *Communications of the ACM* 59(2) (2016), 56–62.

10 A. Etzioni and O. Etzioni, Designing AI systems that obey our laws and values, *Communications of the ACM* 59(9) (2016), 29–31.

11 E. Fast and E. Horvitz, Long-term trends in the public perception of AI, in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

12 S.A. Friedler, C. Scheidegger, S. Venkatasubramanian, S. Choudhary, E.P. Hamilton and D. Roth, A comparative study of fairness-enhancing interventions in machine learning, arXiv:1802.04422, 2018.

13 N. Frosst and G.E. Hinton, Distilling a Neural Network Into a Soft Decision Tree, *CoRR*, abs/1711.09784, 2017.

14 M. Gini, N. Agmon, F. Giunchiglia, S. Koenig and K. Leyton-Brown, AI in 2027, *AI Matters* 4(1) (2018), 10–20.

15 N. Goodall, Ethical decision making during automated vehicle crashes, *Transportation Research Record: Journal of the Transportation Research Board* 2424 (2014), 58–65.

16 I. Goodfellow, Y. Bengio and A. Courville *Deep learning*, Vol. 1, MIT Press, 2016.

17 K. Grace, J. Salvatier, A. Dafoe, B. Zhang and O. Evans, When will AI exceed human performance? Evidence from AI experts, arXiv:1705.08807, 2017.

18 R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar and K. Sycara, Transparency and explanation in deep reinforcement learning neural networks, in *Proc. of the AAAI/ACM Conference on AI, Ethics, and Society*, 2018.

19 M.I. Jordan and T.M. Mitchell, Machine learning: Trends, perspectives, and prospects, *Science* 349(6245) (2015), 255–260.

20 D. Lazer, The rise of the social algorithm, *Science* 348(6239) (2015), 1090.

21 Y. Li, Deep reinforcement learning: An overview, arXiv:1701.07274, 2017.

22 G. Marcus, Deep learning: A critical appraisal, arXiv:1801.00631, 2018.

23 B.D. Mittelstadt, P. Allo, M. Taddeo, S. Wachter and L. Floridi, The ethics of algorithms: Mapping the debate, *Big Data & Society* 3(2), 2016.

24 V.C. Müller and N. Bostrom, Future progress in AI: A survey of expert opinion, in V.C. Müller, ed., *Fundamental Issues of AI*, Springer, 2016, pp. 555–572.

25 N.J. Nilsson, *The Quest for Artificial Intelligence*, CUP, 2009.

26 M.T. Ribeiro, S. Singh and C. Guestrin, Why should I trust you?: Explaining the predictions of any classifier, in *Pr. ACM SIGKDD*, 2016, pp. 1135–1144.

27 N.M. Richards and W.D. Smart, How should the law think about robots?, in R. Calo, A.M. Froomkin and I. Kerr, eds., *Robot Law*, Edward Elgar Publishing, 2016, pp. 3–22.

28 S. Russell, D. Dewey and M. Tegmark, Research priorities for robust and beneficial AI, *AI Magazine* 36(4) (2015), 105–114.

29 T.J. Sejnowski, Learning optimal strategies in complex environments, *Proceedings of the National Academy of Sciences*, 107(47) (2010), 20151–20152.

30 M. Shanahan, *The Technological Singularity*, MIT Press, 2015.

31 D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, *Nature* 529(7587) (2016), 484–489.

32 D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge. *Nature* 550(7676) (2017), 354.

33 S. Steinert, The five robots–a taxonomy for roboethics, *Int. J. of Social Robotics*, 6(2) (2014), 249–260.

34 P. Stone, R. Brooks, E. Brynjolfsson, R. Calo, O. Etzioni, G. Hager, J. Hirschberg, S. Kalyanakrishnan, E. Kamar, S. Kraus, et al., *AI and Life in 2030. One Hundred Year Study on AI*, Report of the 2015–2016 Study Panel,Stanford University, http://ai100.stanford.edu/2016-report, 2016.

35 R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2017, 2nd ed.

36 J. Taylor, E. Yudkowsky, P. LaVictoire and A. Critch, Alignment for advanced machine learning systems, MIRI, 2017 (unpublished) https://tinyurl.com/jemdx4b.

37 M. Tegmark, *Life 3.0. : Being Human in the Age of AI*, Allen Lane, 2017.

38 O. Tene and J. Polonetsky, A theory of creepy: Technology, privacy, and shifting social norms, *Yale Journal of Law and Technology* 16(1), 2014.

39 P. Vamplew, R. Dazeley, C. Foale, S. Firmin, and J. Mummery, Human-aligned AI is a multiobjective problem. *Ethics and Information Technology* 20(1) (2018), 27–40.

40 M. van Otterlo, A machine learning perspective on profiling, in M. Hildebrandt and K. de Vries, eds., *Privacy, Due Process and the Computational Turn*, Routledge, 2013, Chapter 2, pp. 41–64.

41 M. van Otterlo, The libraryness of calculative devices, in L. Amoore and V. Piotukh, eds., *Algorithmic Life: Calculative Devices in the Age of Big Data*, Routledge, 2016, Chapter 2, pp. 35–54.

42 M. van Otterlo, From intended archivists to intentional algivists: Ethical codes for humans and machines in the archives, in F.P. Smit, A. Glaudemans and R. Jonker, eds., *Archives in Liquid Times*, Stichting Archiefpublicaties (S@P), 2017, Chapter 12, pp. 267–293.

43 M. van Otterlo, From algorithmic black boxes to adaptive white boxes: Declarative decision-theoretic ethical programs as codes of ethics, in *Proc. of the AAAI/ACM Conference on AI, Ethics, and Society*, 2018.

44 M. van Otterlo, Gatekeeping algorithms with human ethical bias: The ethics of algorithms in archives, libraries and society, arXiv:1801.01705, 2018.

45 T. Walsh, *Android Dreams: The Past, Present and Future of AI*, Hurst Publishers, 2017.

46 M.A. Wiering and M. van Otterlo, eds., *Reinforcement Learning: State-of-the-Art*, Springer, 2012.

47 See my 'Ethics of algorithms' course, https://tinyurl.com/ydanrvpz.

48 *The Malicious Use of AI*, https://maliciousaireport.com.

49 https://tinyurl.com/ya8fsatf.

50 https://tinyurl.com/mvh4y8s.

51 https://tinyurl.com/ya2v6lct.

52 https://tinyurl.com/le8wlyu.

53 https://tinyurl.com/y89s3ztw.

54 https://tinyurl.com/y89s3ztw.

55 https://tinyurl.com/ybjj4ske.

56 https://tinyurl.com/z7xn9sm.

57 https://tinyurl.com/y9bmnuum.

58 https://tinyurl.com/hyvmpwb.

59 https://tinyurl.com/h3mwshw.

60 https://tinyurl.com/y9b2ujhb.

61 https://tinyurl.com/yd4rwfow.

62 https://www.fatml.org.

63 http://www.aies-conference.com.

64 https://ethicsinaction.ieee.org.

65 https://futureoflife.org/ai-principles.

66 https://tinyurl.com/yb8a5w5c.

67 https://tinyurl.com/ybkhdjhm.

68 https://tinyurl.com/ybc5v798.

69 https://tinyurl.com/jzdpyz5.

70 https://tinyurl.com/y9lqksgf.

71 http://moralmachine.mit.edu.

Nicos J. Starreveld
Korteweg-de Vries Instituut
Universiteit van Amsterdam
n.j.starreveld@uva.nl

Interview Lambert Schomaker

"Het gaat nu echt heel snel"

Op 18 juni 2018 heeft Nicos Starreveld, redacteur van dit blad, prof. dr. Lambert Schomaker geïnterviewd, hoogleraar kunstmatige intelligentie aan de Rijksuniversiteit Groningen.

Op maandag 18 juni ben ik naar Groningen gegaan om prof. dr. Lambert Schomaker te interviewen. Om 10.00 uur stond ik voor zijn deur in het Bernoulliborg-gebouw op de Zernike Campus Groningen. Ik was erg enthousiast en tegelijkertijd zenuwachtig voor het interview. Toen ik naar binnen liep en professor Schomaker sprak, wist ik meteen dat het een leuk gesprek zou worden!

Naar machine learning

Voordat Lambert Schomaker betrokken raakte bij machine learning, als jonge promovendus te Nijmegen, was hij vooral geïnteresseerd in fysiologie. In zijn afstudeerproject deed hij onderzoek naar elektrofysiologie en naar de modellering van spieractiviteiten, een onderwerp dat later een belangrijke rol zou gaan spelen in zijn promotieonderzoek over handschriftherkenning.

Wat vond u zo interessant aan fysiologie?

"In die tijd, de jaren tachtig, waren er al computermodellen ontwikkeld van spieractiviteit. Wat ik ontzettend interessant vond was die koppeling tussen een natuurlijk fenomeen en een wiskundig model, hoe je alle elektrische activiteit van de spieren kunt optellen en dan een uitspraak kunt doen over onder andere het spectrum van spieractiviteit."

En de stap naar machine learning?

"Toen ik klaar was met mijn masteropleiding en naar een baan zocht, waren er geen vacatures op het gebied van fysiologische psychologie. De vacature in Nijmegen betrof cognitieve psychologie (*cognitive science*) en ging over het modelleren van handschriften met computers."

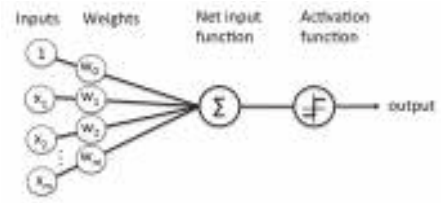
En was het mogelijk om uw kennis van uw eerdere opleiding in dit project te gebruiken?

"Het idee achter mijn onderzoeksproject was dat je een handschrift neemt van een willekeurig persoon en dan gaat kijken of de computer een nieuw handschrift kan genereren in dezelfde stijl. Het idee was ook dat je terwijl je iets bouwt je iets moet leren over het onderliggende fenomeen, de onderliggende processen, iets dat we in ons vakgebied heel belangrijk vinden, we noemen het *understanding by building*. [...] In het Nijmeegse instituut voor cognitie en informatica hing een levendige sfeer. Het was een multidisciplinair instituut, met mensen uit verschillende vakgebieden, waaronder wiskunde, informatica en psychologie."

Lambert Schomaker

Korte biografie

Prof. dr. Lambert Schomaker, geboren te Goirle in 1957, is in 1983 afgestudeerd in fysiologische psychologie in Tilburg. In 1991 is hij in Nijmegen gepromoveerd met als promotieonderwerp 'simulatie en herkenning van handschrift'. Hij is sinds 2001 hoogleraar kunstmatige intelligentie aan de Rijksuniversiteit Groningen met als expertise machinaal leren (*machine learning*), kunstmatige intelligentie, patroonherkenning en cybernetica.



Figuur 1 Rosenblatts perceptron.

En hoe zat het in die tijd met machine learning?

"In de jaren tachtig praatten we over de tweede golf van neurale netwerken. De eerste golf was in 1950–1960 met Rosenblatts perceptron (zie Figuur 1). Daar zaten beperkingen in en er kwam ook een wiskundig bewijs van Minsky dat zo'n perceptron niet alle problemen aankon. Midden jaren tachtig kwam de tweede golf waarin de neurale netwerken ingewikkelder gemaakt konden worden en ze complexe problemen aankonden. Toen kreeg je rond 1996 een piek in publicaties, daarna stortte het eventjes in, maar vanaf 2005 is er weer een hele opleving van neurale netwerken. Een enorme revolutie van machine learning met neurale netwerken."

Promotieonderzoek – handschriftherkenning

In zijn promotietraject heeft Lambert Schomaker onderzoek gedaan in machine learning en de mogelijke toepassingen daarvan in handschriftherkenning en computersimulatie van motoriek. Het doel van zijn promotieproject was om een computermodel te ontwikkelen dat in staat zou zijn om handschriften te genereren. Een observatie die essentieel is voor zijn onderzoek, is dat het handschrift ruimtelijke en tijdelijke karakteristieken heeft. Elke schrijver bezit zijn eigen schrijfstijl. Er is dus variatie in schrijfstijl tussen verschillende schrijvers, maar tegelijkertijd is er ook variatie in het geschrift van elke schrijver afzonderlijk. Schomaker was daarom begonnen met het bestuderen van de fysiologische processen die betrokken zijn bij het menselijke schrijfproces. Om de belangstelling in het onderliggende schrijfproces te kunnen begrijpen, moeten we teruggaan in de tijd, naar de jaren vijftig, toen de eerste algoritmen van patroon- en beeldherkenning werden ontwikkeld. Men begon toen de rekenkracht van de nieuw ontwikkelde computers te benutten. De eerste algoritmes van patroonherkenning waren gebaseerd op een logische-symbolische benadering. Bij elk individueel beeldelement ('pixel') werd een binaire waarde toegekend en vervolgens werd die opgevat als een logische propositie. Twee beelden konden dus met elkaar vergeleken worden door de corresponderende logische proposities te vergelijken. Deze algoritmes konden wel succesvol machinale drukletters herkennen, maar de automatische herkenning van handgeschreven tekst in een vrije stijl bleek echter een bijzondere uitdaging te zijn [5]. Dat ligt voornamelijk aan het feit dat er een grote variatie zit in het geschrift van een specifieke schrijver. Schomaker vat het samen in zijn oratie:

> "De complexiteit van de woordbeschrijvingen voor alle mogelijke stijlen is fenomenaal en computationele belasting van het doorzoeken van de grote verzameling symbolische structuren wordt al snel te groot, ook voor de computer van vandaag (2001) en volgend jaar. De menselijke patroongenerator blijkt in staat te zijn tot een bijna oneindige variatie van vormen in het platte vlak."

Het bleek dus dat een andere benadering dan de methoden uit de logica nodig was. Men begon gereedschappen uit andere wiskundige disciplines te gebruiken, zoals de statistiek, de meetkunde, de lineaire algebra en de signaaltheorie.

"In de jaren tachtig ontstond er een stroomversnelling in de patroonherkenning door de verdere ontwikkeling van Markovmodellen, neurale netwerken en Bayensiaanse methoden. In essentie gaat het erom dat deze methoden, expliciet of impliciet, statistisch van aard zijn. In plaats van het opdringen van een op symbolische manipulatie gebaseerd formalisme, is het uitgangspunt het volgende: indien er sprake is van regelmaat in gegevens, dan moet er een algoritme bestaan dat deze regelmaat kan detecteren. [...] Uitgaande van een in de Nijmeegse Schrijfonderzoeksgroep ontwikkelde theorie over het schrijfproces, kan men de beweging van de penpunt onderverdelen in halen. [...]"

Figuur 2 Het woord algebra, geschreven door acht schrijvers (rijen), ieder vier keer (kolommen).

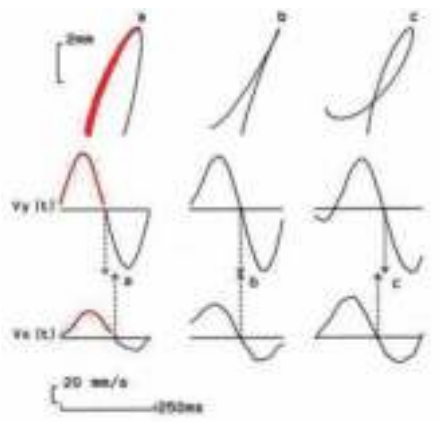
Figuur 3 Een woord is vaak niet meer dan een krabbel inkt.

In zijn promotieonderzoek had Schomaker bedacht dat het herkennen van afzonderlijke letters in een handschrift niet zal werken: “Een geschreven woord is vaak niet meer dan een krabbel inkt (zie Figuur 3) zonder duidelijk herkenbare letters. Deze krabbel bevat echter genoeg informatie voor digitale herkenning van het hele woord.” [4]

Laten we dit in het volgende voorbeeld uit Schomakers oratie bezien:

> “Spectraalanalyse wijst op een sterke periodiciteit van de beweging, van 5 Hertz. Berekening van de gemiddelde schrijfduur per haal wijst uit dat een enkelvoudige beweging in schrift wordt begrensd door minima in de schrijfsnelheid. Een modale haal in het schrijfproces duurt ongeveer 100 milliseconden. Het verticale snelheidspatroon van twee opeenvolgende halen benadert een volledige periode van een sinusoïde.”

Wat waren de grootste ontwikkelingen in machine learning die een invloed hadden op uw promotieonderzoek?
“Die periode was de periode waarin kunstmatige intelligentie langzaam overging van het symbolische denken — wat voor computers heel nuttig was maar voor *computer vision* vrijwel onbruikbaar — naar neurale netwerken van de tweede generatie. In mijn proefschrift zitten inderdaad elementen van de traditionele symbolische benadering, maar ik was van de generatie die meteen heel enthousiast was over die nieuwe neurale netwerken.”



Figuur 4 Halendefinitie op basis van snelheid en faseverschil tussen horizontale en verticale snelheid. De drie hoofdvormen zijn hier gegeven.

Onderzoek in machine learning
Na zijn promotie en tot 2001 had Schomaker een onderzoeksaanstelling in Nijmegen. In 2001 is hij hoogleraar kunstmatige intelligentie in Groningen geworden. Hij is projectcoördinator geweest van verschillende Europese projecten over machine learning, patroon- en handschriftherkenning en robotica. Zijn aandacht lag, en ligt nog steeds, voornamelijk bij neurale netwerken.

We horen vaak dat neurale netwerken niet transparant zijn en dat we niet precies kunnen weten hoe ze tot een specifieke uitkomst zijn gekomen. Ze worden vaak een ‘black box’ genoemd! Is dat een grote beperking?
“Dat is een vraag die vaak naar voren kwam bij lezingen en conferenties, er was een verwijt over neurale netwerken dat ze niet transparant waren of dat je niet begreep wat ze deden. Ik denk dat het belangrijk is in zo’n systeem, zoals een neuraal netwerk, dat het zich voorspelbaar gedraagt en dat je dus de statistiek begrijpt van de antwoorden van het systeem, dat je daarin vertrouwen kunt hebben. Er zijn een heleboel instrumenten in de wereld die eigenlijk niet heel goed begrepen worden, maar die gebruikt worden omdat de statistiek van de meting heel goed bekend is. Het voorbeeld dat ik graag altijd gebruik is de bloeddrukmeting. Als je bloeddruk wilt meten, denk ik dat je een naald in een ader moet steken en daar een sensor aan moet verbinden en de meting dan moet omschalen naar de echte druk. Maar deze invasieve methode mag dan wel exact zijn, hij is ook gevaarlijk. Dus er is een uitvinding gedaan dat als je de biceps met een band afklemt en je luistert met een stethoscoop, dat je dan langzaam maar zeker hoort dat het bloed wordt afgeknepen en dan ontstaat er een idee van onderdruk en bovendruk. Dat is volgens mij helemaal niet zo’n transparant en duidelijk systeem! Maar de methode is niet invasief, wordt geaccepteerd, en vervolgens verzamel je je statistiek en je krijgt er vertrouwen in. En ik denk dat dit ook voor neurale netwerken geldt.”

U sprak over de statistiek en de betrouwbaarheid van de uitkomsten van een model. Wat voor gereedschap gebruikt men om dat te doen?
“Er zijn verschillende methodes. De eerste, die ik al genoemd heb, is dat je kijkt naar een heleboel datasets en dat je dan, ook door de tijd heen, in de gaten blijft houden wat fout is, wat je systeem levert en wat je had verwacht. Maar je wil soms inderdaad iets meer begrijpen. Wat we gebruiken als methode is dat je uit een werkend neuraal netwerk metingen neemt en daar de statistiek van gaat bepalen, om te snappen wat het netwerk heeft geleerd. En dan kun je alle statistische methodes toepassen op de interne representaties van het neurale netwerk. Soms komen er hele interessante resultaten uit, en in principe is die manier van analyseren al bekend van de tweede golf van neurale netwerken. Dus als je dat wilt dan is het helemaal niet zo’n black box, maar dan kun je begrijpen wat er aan de hand is.”

En presteert een neuraal netwerk beter dan een systeem waarin alle regels expliciet bepaald worden en je daardoor precies kunt weten wat het systeem doet?
“Als je een ingewikkeld systeem hebt, dan zitten er zoveel regels in dat geen mens in redelijkheid snapt wat er aan de hand is. Een complex systeem kan ongeveer tienduizend regels bevatten en is het onderhoud van al deze regels dus een enorme klus. Ik denk, en ik vind dat belangrijk, dat als je voor een neuraal netwerk een goed onderbouwde statistiek hebt, dus welke getallen je kunt verwachten met welke kansen, dan heb je een kwalificatie van het systeem. Als je een systeem expliciet probeert uit te schrijven dan is het veel te data-intensief.”

Forensisch onderzoek en de Dode Zeerollen
Handschriftanalyse is van groot belang bij forensisch onderzoek en bij het bestuderen van historische handschriften. Schomaker heeft verschillende projecten uitgevoerd, promovendi begeleid en een significante bijdrage geleverd in beide gebieden.

Bij het forensisch onderzoek is handschriftanalyse vaak gebruikt bij het oplossen van misdaden. Er is bijvoorbeeld een miltvuurbrief ontvangen met een handgeschreven adres, of een brief waarin losgeld wordt geëist. Vervolgens wil de politie natuurlijk weten of dit geschreven is door

Figuur 5 Forensisch onderzoek naar handschriften.

een eventuele verdachte. Daarbij gebruiken ze een forensisch expert. Die vergelijkt het handschrift met dat van de verdachte en zegt vervolgens of het gaat om dezelfde persoon. Maar dit is een subjectief proces. Bovendien kan de expert bevooroordeeld zijn. Lambert Schomaker en Marius Bulacu, gepromoveerd onder begeleiding van Schomaker, hebben software ontwikkeld die gebruikt kan worden om tot een overtuigende conclusie te kunnen komen. Met hun software heeft de forensisch expert een hulpmiddel om zijn oordeel te onderbouwen [1, 2, 6]!

De Dode Zeerollen behoren tot de belangrijkste archeologische vondsten ooit. Halverwege de vorige eeuw werden in de grotten rond Qumran bijna duizend manuscripten ontdekt uit de tijd vóór en rond het begin van onze jaartelling. Ze bieden een unieke inkijk in het ontstaan van wat later de Bijbel is gaan heten. "De inhoud van de rollen weten we," vertelt Mladen Popović, directeur van het wereldvermaarde Qumran Instituut van de Rijksuniversiteit Groningen, "maar we willen ook de wereld achter de geschriften kennen. Wie schreef welke tekst of tekstdelen? En hoe zat die schrijverscultuur in elkaar? Daarmee komen we in zekere zin bij de bronnen van onze cultuur." Om antwoord te vinden op zijn vragen heeft hij met Lambert Schomaker samengewerkt. In het Artificial Intelligence Institute van de RUG is een systeem ontwikkeld voor de ontsluiting van historische en handgeschreven archieven: MONK. Ook alle fragmenten van de Dode Zeerollen zijn ingevoerd. MONK moet nu leren letters, woorden en handschriften te herkennen. Popović: "MONK ziet meer dan je met het oog kunt zien. Het systeem houdt bijvoorbeeld ook rekening met spierkracht, penvoering, het materiaal dat gebruikt is. Op die manier kun je de unieke kenmerken van een schrijver op het spoor komen." Zo wordt het mogelijk te achterhalen wie welke teksten schreef, of er één of meerdere mensen aan een tekst werkten en hoe belangrijk hun rol was. Zo is het mogelijk de Dode Zeerollen en de cultuur waar zij uit voortkwamen nog beter te begrijpen [3, 6].

Van ML naar kunstmatige intelligentie

Voordat ik Lambert Schomaker had gesproken, wist ik al dat er nog steeds veel uitdagingen en open problemen voor de hand liggen om over intelligente systemen te kunnen spreken. Afgelopen vijftig jaar heeft men grotendeels met twijfel gereageerd op de vraag of het mogelijk is om een intelligent systeem te bouwen. Ook binnen Nederland heeft het lang geduurd voordat kunstmatige intelligentie een prominente plek kon krijgen binnen de wetenschap, voornamelijk vanwege de twijfels rondom de toekomst en het karakter van kunstmatige intelligentie. Onder anderen ook de informaticus Edsger Dijkstra, winnaar van de prestigieuze Turing-prijs, heeft vaak zijn mening daarover geuit. In een verslag, geschreven toen hij hoogleraar in Austin was, lezen we het volgende:

Figuur 6 Fragmenten van de Dode Zeerollen.

> "The Fathers of the field had been pretty confusing: John von Neumann speculated about computers and the human brain in analogies sufficiently wild to be worthy of a medieval thinker and Alan M. Turing thought about criteria to settle the question of whether Machines Can Think, a question of which we now know that it is about as relevant as the question of whether Submarines Can Swim."

Dat schreef Edsger Dijkstra in 1984. Nu, anno 2018, stel ik aan Lambert Schomaker precies dezelfde vraag.

We hebben het lang over machine learning gehad maar hoe ver zijn we eigenlijk van een systeem dat echt kan leren?

Schomakers visie op de toekomst van machine learning is voornamelijk gebaseerd op twee kernprincipes: het leerproces van het neurale netwerk moet dynamisch en zelfstandig worden!

"Het gaat nu echt heel snel! We hebben Go gehad, dat aanvankelijk in de eerste opzet werd getraind op basis van expert games. Toen kwam er kritiek uit ons eigen vakgebied, maar ook uit het algemeen publiek: het is inderdaad geen AI, het is in feite *intelligence by proxy*, je hebt het systeem getraind met menselijke voorbeelden. Dat hebben ze zich bij DeepMind aangetrokken, en ze zijn terug gegaan naar AlphaGo Zero en toen is het systeem opnieuw getraind met alleen de regels van het spel, en ook dan was het systeem beter. Hij heeft wel heel vaak tegen zichzelf gespeeld, in een simulatie kan het systeem tegen zichzelf spelen en de gewichten van zo'n neuraal netwerk worden geüpdatet."

Leert AlphaZero ook tijdens het spel?

"Nee, dat is daar helemaal uitgeschakeld. Daar is ook een scheiding tussen een laboratoriumfase en een operationele fase. Het is een statisch systeem! Ik vind het schakelen naar een dynamisch leerproces met *constant learning* een van de belangrijke stappen die we moeten maken in de toekomst. En dat staat in mijn agenda om te proberen!"

En hoe worden tegenwoordig neurale netwerken in de praktijk getraind?

"Wat men tegenwoordig doet is een dataset opsplitsen in twee delen, een deel van de data wordt gebruikt om een probleem

Figuur 7 Beeld uit AlphaGo Zero-promo van DeepMind.

te leren en het tweede deel wordt gebruikt om het systeem te testen. De dataset wordt ook vaak gekozen zodat die stationair en gebalanceerd is. Maar dat is geen AI! Mijn kritiek op dat paradigma is dat een systeem dat op deze manier getraind is, zelden werkt in de echte wereld, als er echte data komt, die vaak niet stationair is en vaak ruis bevat. AI is dat het systeem zelf in de gaten houdt wanneer het aangepast moet worden, dat het de veranderingen in de echte data kan detecteren en vervolgens zelf de geleerde patronen kan aanpassen. Daar zijn technieken voor, bijvoorbeeld *concept drift* om er een te noemen. Ik vind het realistischer om toe te gaan naar systemen die in staat zijn om over een langere periode de verandering van het onderliggende systeem op te pakken."

Schomaker vervolgt: "Hier wil ik nog iets aan toevoegen waar we het nog niet over hebben gehad: de leervoorbeelden. Dus overal waar je veel voorbeelden hebt, heb je enorm succes, waar je geen voorbeelden hebt is het een stuk lastiger. *Unsupervised learning* noemen we dat, waarbij het systeem alleen naar de data kijkt zonder dat het afhankelijk is van een target of labeled dataset. Een target dataset zegt wat het systeem gedaan zou moeten hebben. Maar om die gelabelde data te verzamelen is in sommige toepassingen erg kostbaar! We zijn in een overgangsfase waarin de kunstmatige systemen heel veel lenen van de menselijke feedback."

Tijdens mijn gesprek met Lambert Schomaker realiseerde ik me dat machine learning een alternatieve methode aanbiedt om de wereld rond ons te kunnen analyseren, een alternatieve methode om fenomenen te kunnen bestuderen.

"Wat je ziet is dat de manier waarop ik zelf ben opgeleid destijds is dat je naar een fenomeen kijkt, en dan moet je het probleem snappen, je moet het abstraheren, en je moet het modelleren en dan heb je het probleem in de greep. En daarover is veel te zeggen. Als je de middelen hebt om dat te doen, dan zou ik zeggen probeer het en als dat je lukt dan is dat enorm bevredigend. Maar is dat altijd wat je wilt? Is dat altijd begrijpen wat er nodig is?"

Deze laatste vraag heb ik meegenomen naar huis en erover nagedacht. Laten we even teruggaan naar het voorbeeld met de meting van je bloeddruk. Je hoeft eigenlijk niet in alle detail te weten hoe je lichaam werkt om je bloeddruk te meten en vertrouwen te hebben in die meting! Zolang er een betrouwbare statistiek bestaat kun je die gebruiken om iets te zeggen over je gezondheid!

"In een industriële setting is de prestatie van een systeem toch eigenlijk belangrijker dan het begrip. Ik denk dat waar het kan je moet proberen het fenomeen expliciet te modelleren, maar is het toch ook niet zo dat het wiskundig gezien juist heel interessant is om na te denken over de eigenschappen waar problemen aan moeten voldoen zodat ze automatisch geschat kunnen worden! Dus dan richt je je wiskunde niet meer op het modelleren van het fenomeen in de buitenwereld, maar die richt je dus op de fundamentele wiskundige eigenschappen van problemen die te schatten zijn, die op te lossen zijn met een automatisch rekenproces. En dat is toch wiskundig gezien een fantastische uitdaging! Ik vind het een hele leuke tijd waarin er zoveel successen zijn en zoveel nieuwe vergezichten aandienen!"

Met deze boodschap voor de toekomst wil ik dit, voor mij erg leerzame en inspirerende, interview afsluiten. Ik wil professor Lambert Schomaker bedanken voor de prettige discussie!

Referenties

1 A.A. Brink, *Robust and Applicable Handwriting Biometrics*, proefschrift, RUG, 2011.

2 M.L. Bulacu, *Statistical Pattern Recognition for Automatic Writer Identification and Verification*, proefschrift, Artificial Intelligence Institute, RUG, 2007.

3 M. Dhali, S. He, M. Popovic, E. Tigchelaar en L. Schomaker, A digital palaeographic approach towards writer identification in the Dead Sea Scrolls, in *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods*, Vol. 1, 2017, pp. 693–702.

4 L. Schomaker, *Simulation and Recognition of Handwritten Movements*, proefschrift, 1991.

5 L. Schomaker, Patronen en symbolen: een wereld door het oog van de machine, oratie, 2002.

6 rug.nl/nieuws.

Casper Albers
Psychometrie & Statistiek
Rijksuniversiteit Groningen
c.j.albers@rug.nl

Column Casper sees a chance

The statistical approach known as Machine Learning

This issue of the *Nieuw Archief voor Wiskunde* is devoted to Machine Learning. Although this has a very modern sounding name, the majority of techniques in machine learning are borrowed from other fields, most notably statistics, and date back many decades. Yet, there's more to machine learning than simply copying other fields' work.

The field of machine learning has emerged from computing science and, since about thirty years, it has been recognised as a separate branch of science. As such, it is a young and booming field with terms such as '(un)supervised learning', 'Bayesian networks', 'clustering methods' and 'classification trees'. To a large extent, these are new names for old concepts.

Historically, statistics is the scientific field of studying quantitative information under uncertainty. As such it emerged from mathematics, with its roots dating back to renaissance scientists such as Blaise Pascal, Christian Huygens and Jakob Bernoulli. What all statistical methods have in common, whether it concerns the elementary computation of a mean or a complicated statistical test for a high-dimensional data set, is that they aim to extract information from raw data. With the amount of data growing over time, and the emergence of computers in the second half of the previous centuries, statistical methods became more and more computationally intensive. Already in 1977, the International Statistical Association founded the International Association for Statistical Computing. Thus, doing data analysis with computers is not a novelty invented by machine learning.

Same concepts, different words

Stanford statistics professors Robert Tibshiriani, Jerome Friedman and Trevor Hastie — three of the very few that managed to obtain rockstar status among both statisticians and machine learners — offer a course in machine learning. As part of their course material, they provide a table comparing terms in both fields, part of which is reproduced here [5]:

Machine learning	Statistics
Network, graph	Model
Weights	Parameters
Learning	Fitting
Generalization	Test set performance
Supervised learning	Regression, classification
Unsupervised learning	Density estimation, clustering

Photo: James Jowers / George Eastman House Collection

There is some tongue-in-cheek in this table, but they do hit a nerve. Indeed, a basic supervised learning model is nothing more than a linear regression model without a check on the validity of the underlying assumptions.

Statistics

Statistical methods can be roughly subdivided into two disciplines: estimation and inference. Estimation techniques are used to reduce a (relatively) large amount of quantitative information, into a smaller and more comprehensible amount of information. Examples are the mean, standard deviation and visualisations such as a histogram. On the other hand, inferential models form a structured way to use data to infer about two competing hypotheses and, often, decide between two competing actions. An example is the testing of the efficacy of a new drug, by comparing the health of a treatment group to that of the placebo group. Should the difference be sufficiently large, then the statistician would argue that it is so unlikely that such a difference (or even larger) would occur by chance, that it is safe to reject the hypothesis that the drug has no beneficial health effect. Usually such a model relies on a set of assumptions, such as a linear relation between age and health, the same amount of health fluctuations between men and women, normally distributed residuals, et cetera. To build such a model, expert knowledge is required: medical experts can tell you that it is sensible to allow for gender differences when studying health, but that allowing for differences based on the number of letters in your first name is less sensible. Together with the expert, the statistician builds the model.

The goal of statistics is understanding the data-generating process, with the aim to make better decisions. However, the model only 'works' if the underlying assumptions are valid. Checking model validity thus is a major part of statistical analyses. Some non-parametric techniques exist that make fewer assumptions (e.g. not assuming normality), but models that make no assumption whatsoever are very rare in statistics.

As George Box said: "All models are wrong, but some models are useful." The statistician knows that her/his model is a simplification of reality. However, just as the mean and standard information provide less information than the full sample, the statistician is willing to accept this: there is a trade-off between model complexity and model comprehensibility (and, thus, usability). If, for instance, the relation between age and health is not linear (and I'm sure it isn't), but also not too far from linear, then assuming linearity greatly increases the comprehensibility of the model.

Machine learning

This is where the main difference between statistics and machine learning pops up: machine learning models are fully assumption-free. Furthermore, the machine learner does not care about model validity: the main (and often only) point of interest is prediction accuracy.

Machine learners can make assumption free models by relying on the bootstrap and cross-validation, tools developed by another cross-discipline rockstar, Bradley Efron. A related, common, method is to split the total sample into a training and validation set. The machine learning model is given the training data, e.g. a list of people's age, gender, health and whether or not they are part of the treatment group. The algorithm then searches this data set for patterns. If, indeed, older people are of lower health, the algorithm will learn this — without the restriction that the relation must be linear. As such, the supervised learning algorithm is nothing more

or less than statistical non-parametric, non-linear regression, yet with a different objective: comprehensibility is irrelevant (although some people in machine learning do focus on this). It is perfectly fine if the algorithm is a black box, as long as it learns the relations between the variables as good as possible. The algorithm thus creates some kind of procedure that, given someone's age, gender and drug use information, predicts this person's health as good as possible, without revealing how the exact relation between the variables is.

This algorithm is subsequently applied to the validation set, and the predictions are compared with the actual health scores. A value like the mean squared prediction error provided information on the model performance. As the predictions are made without a model, the machine learner doesn't have to worry about model validity. This advantage comes at the cost that the machine learner also doesn't learn about the data generating process.

Different goals

Thus, many of the techniques used in statistics and machine learning are more or less equivalent. Yet, the purpose of their use, and the chosen strategy, is fundamentally different. Whereas statistics is concerned with understanding processes and providing tools for informed decision making, (most fields in) machine learning is concerned with predicting future observations as good as possible.

In some situations, learning about the data generating mechanisms actually is the core scientific question, in which case statistical models are essential. In other cases, e.g. predicting where traffic queues will occur or whether some stock price will go up or down, a good prediction might be more valuable than a good explanation. Which of the two mindsets is most useful thus depends on the nature of the question you want answered. This is also argued by Leo Breiman [1], in a thought-provoking paper with commentaries from the likes of Sir David Cox and Bradley Efron.

Fawcett and Hardin [3] give a nice metaphor describing the relation between both fields: "[Statistics and machine learning] are like two pairs of old men sitting in a park playing two different board games. Both games use the same type of board and the same set of pieces, but each plays by different rules and has a different goal because the games are fundamentally different. Each pair looks at the other's board with bemusement and thinks they're not very good at the game."

Marketing

According to O'Connor [2], the reason that machine learning invited all these new terms for existing concepts, and has become hugely popular in doing so, is that statistics has a marketing problem: "Machine learning sounds like it's young, vibrant, interesting to learn, and growing; statistics does not." He cites Friedman [4] who states: "One can catalog a long history of Statistics (as a field) ignoring useful methodology developed in other data related fields. Here are some of them [...] that had seminal beginnings in statistics but for the most part were subsequently ignored in our field: pattern recognition, neural networks, machine learning, graphical models/Bayesian networks, chemometrics, data visualization. [...] One view says that our field should concentrate on that small part of information science that we do best, namely probabilistic inference based on mathematics. If this view is adopted, we should become resigned to the fact that the role of Statistics as a player in the 'information revolution' will steadily diminish over time."

Friedman's message is clear: that other fields seemingly took over large parts of data analysis, is statistics' own fault. The field should have embraced computational methodology as fundamental statistical tool. Friedman wrote his piece two decades ago and, fortunately, computational methodology is playing a more relevant role in statistics, most notably in applied statistics, than at the turn of the century. The change has come to late to claim all computational data science to be part of statistics — that ship has sailed. Friedman also paraphrases Efron by stating: "Those who ignore statistics are condemned to reinvent it." At least for the near future, the fields of statistics and machine learning will have to co-exist and co-operate together.

Conclusion

To summarise, I have outlined that many machine learning-ideas have been 'borrowed' from statistics. Some statisticians experience frustration or jealousy from this. Although somewhat understandable, it's like that kid in school that got popular by stealing your joke, I do not share this emotion.

In my experience, teaching statistical thinking, i.e. reasoning under uncertainty, requires different strategies for different audiences. An undergraduate student in mathematics can be taught that the ordinary least squares estimator is 'the optimal' estimator in regression by walking her/him through the elegant proof of the Gauss–Markov theorem. Students in social sciences, however, prefer heuristic explanations based on accessible simulations. I can very well imagine that students with a solid background in computing, will benefit from an explanation in terms of algorithms and other common concepts.

In the end, what matters is that as many students learn how to perform a scientifically validated analysis of quantitative data and, as such. What label they assign to their approach, whether it is statistics, machine learning, or some umbrella term as data science, is virtually irrelevant. If machine learning helps to get more people to do solid data analysis, I'm on board. In this age of fake news and fake information, all scientifically based methods for turning data into comprehensible information should be welcomed with open arms.

References

1 Leo Breiman, Statistical modeling: The two cultures (with discussion), *Statistical Science* 16(3) (2001), 199–231, projecteuclid.org/euclid.ss/1009213726.

2 Brendan O'Connor, Statistics vs. machine learning: Fight! (2008), brenocon.com/blog/2008/12/statistics-vs-machine-learning-fight.

3 Tom Fawcett and Drew Hardin, Machine learning vs. statistics (2017), svds.com/machine-learning-vs-statistics.

4 Jerome Friedman, Data mining and statistics: What's the connection? *Proceedings of the 29th Symposium on the Interface Between Computer Science and Statistics* (1997).

5 Robert Tibshiriani, Glossary, statweb.stanford.edu/~tibs/stat315a/glossary.pdf.

Jaap van den Herik
Leiden Centre of Data Science
Universiteit Leiden
h.j.vandenherik@law.leidenuniv.nl

Geschiedenis

Computerschaak: van idee tot DeepMind

Het afgelopen jaar versloeg het computerprogramma AlphaGo voor het eerst een wereldkampioen Go. Een paar maanden later kwam DeepMind (Google) met het op deep learning gebaseerde AlphaGo Zero, dat nog veel sterker bleek: AlphaGo Zero versloeg AlphaGo met 100–0. In het schaken was de computer al eerder de mens de baas, maar ook hier zorgde deep learning voor een doorbraak. AlphaZero, de schaaversie van AlphaGo Zero, versloeg het programma Stockfish met 28-0 bij 72 remises. In dit artikel geeft Jaap van den Herik, hoogleraar informatica en recht aan de Universiteit Leiden, een historisch overzicht van de ontwikkelingen in het computerschaak. Hij is al decennialang een leidende figuur op dit gebied.

Dit artikel geeft een overzicht van de ontwikkeling van het computerschaak. Wat is er nodig geweest om de menselijke wereldkampioen te verslaan? Hoe zit het met *intuïtie*? Wat zijn de grote doorbraken geweest? En welke toponderzoekers hebben (veel) van hun tijd besteed aan het computerschaak? Het zijn allemaal interessante onderwerpen en er is maar een beperkte ruimte om ze te bespreken. Voor een goed overzicht heb ik de chronologie genomen en daarbinnen getracht de belangrijkste items in het volle daglicht te stellen aan de hand van de hierboven gestelde vragen.

Illustratie: Wikipedia

Zelfportret van Johann Wolfgang von Kempelen (1734–1804), die als Hofrat van Maria Theresia van Oostenrijk verantwoordelijk was voor de wederopbouw van het door oorlogsgeweld en natuurrampen verwoeste Banat, een regio rond Timisoara. Hij was ook geïnteresseerd in machines en bouwde niet alleen een schaakautomaat, maar ook een machine die de menselijk stem imiteerde.

1770–1940 De eerste schaakmachines

Door de eeuwen heen heeft computerschaak de fantasie geprikkeld. "I will invent a machine for a more compelling spectacle within half a year", beloofde Baron von Kempelen aan keizerin Maria Theresia na een demonstratie van Pelletier in 1769 over magnetisme. Duidelijk is dat von Kempelen de voordracht niet erg inspirerend vond. Het is niet duidelijk uit de overleveringen of hij toen al aan een schaakcomputer dacht of dat hij die in de loop van de hem gegeven tijd (een half jaar) heeft bedacht. Dat de eerste 'machine', die 'De Turk' werd genoemd, een mens bevatte doet er even niet toe. Het idee was geboren en veroverde spoedig de wereld. Het duurde tot 1834 voordat het bedrog uitkwam. Edgar Allen Poe bracht het in 1836 in de openbaarheid. Toch had het idee zoveel aantrekkingskracht dat er in 1868 een nieuwe schaakmachine werd geïntroduceerd onder de naam 'Ajeeb' en twee jaar later nog een onder de naam 'Mephisto'.

Intussen ontwikkelde Babbage (1791–1871) zijn *difference engine* en *analytical engine*. Van de laatstgenoemde dacht hij dat bij een goede implementatie de machine zou kunnen schaken. Babbage beschreef vervolgens een Tic-tac-toe-spelend programma dat hij zelf nooit draaiend heeft gekregen.

Een echte belangrijke bijdrage werd geleverd door Torres y Quevedo (1852–1936). Hij ontwikkelde omstreeks 1890 een elektromechanische machine die in staat was om het eindspel Koning en Toren tegen Koning te spelen. In de beginstelling stond de witte koning op a8 en de witte toren op b7. De zwarte koning mocht overal staan. In 1914 verbeterde hij het mechanisme aanzienlijk. In 1953 implementeerde W. L. (Wim) van der Poel de algoritme in een computerprogramma. Bij een demonstratie van het

programma vond Max Euwe een fout in de algoritme waardoor het programma meer dan vijftig zetten nodig had.

Toen verscheen von Neumann (1903–1957) op het toneel. Zijn publicatie uit 1928 'Zur Theorie der Gesellschaftspiele' bevat de basis voor de minimax-methode. Het werk kreeg echter pas grote bekendheid toen het in samenwerking met Morgenstern werd gepubliceerd in het boek *Theory of Games and Economic Behavior* in 1944.

Het idee om een computer in te zetten bij het nemen van beslissingen is afkomstig van de Duitse ingenieur Konrad Zuse (1910–1995). In 1934 ontwikkelde hij daarover zijn eerste gedachten. Vervolgens bouwde hij in de periode 1935–1937 de eerste relais-rekenautomaat die door een programma bestuurd werd. Zuses Z3-computer die in 1941 operationeel was, wordt tegenwoordig erkend als Turing-compleet en is daarmee de eerste digitale computer. (Let wel, het eerste computerontwerp wordt toegeschreven aan John Vincent Anatanasoff.) Het belang van Zuse voor het computerschaak is gelegen in het ontwerp van de programmeertaal Plankalkül. Hij schreef daarover: "Um klarer zu sehen, wählte ich, u.a., das Schachspiel als Modellfall aus und träumte davon, eines Tages den Schachweltmeister mit einem Computer zu besiegen."

1940–1950 Schaken in Princeton en Bletchley Park

De eerste helft van dit decennium stond in het teken van de tweede wereldoorlog. Geen tijd voor schaken, of toch? De Engelse regering had in Bletchley een onderzoekscentrum gecreëerd met als voorwaarde voor toelating slim en creatief zijn. Dat een relatie met schaken een impliciete aanbeveling was bleek uit de lijst van bewoners van hut 8: C.H.O. d' Alexander, A.M. Turing, I.J. Good, H. Golombek, P. Hilton en D. Michie (er waren er meer). Als er niet aan code-breaking werd gedaan werd er gebrainstormd over de vraag: hoe zou een computer kunnen schaken? Jack Good gaf later het volgende toe: "This procedure [minimax met evaluatie] seemed obvious to us long before Shannon's paper, and I made the mistake of thinking it was not worth publishing." Zo kon het gebeuren dat Turing (1912–1954) meer aandacht aan andere zaken besteedde, ook na de oorlog. Turing en Shannon (1916–2001) brachten

Claude Shannon laat zijn zelfgemaakte elektronische schaakautomaat zien aan schaakmeester Edward Lasker.

beiden een jaar in Princeton door als gast van John von Neumann. Ze hebben zelfs gelijktijdig op Bell Laboratories gewerkt. Daar spraken zij elkaar waarschijnlijk een keer of vijftien, twintig tijdens de lunch, aldus Shannon, maar ze werkten op verschillende afdelingen en mochten niet over hun codewerk spreken omdat dat geheim was (Amerika versus Engeland). Ook over computerschaak hadden ze nooit diepgaand van gedachten gewisseld. Shannon herinnerde zich dat hun gesprekken meestal tot onderwerp hadden de relatie 'human brain–mechanical brain'. Shannons idee was: "Our brain is basically a machine, although it is complicate to simulate."

Shannon was de eerste. Hij had een lezing over computerschaak bij Bell Labs in 1949 en publiceerde zijn beroemde artikel 'Programming a computer for playing chess' in *Philosophical Magazine* (waar anders in die tijd?) in 1950. Het bevatte een beschrijving van drie strategieën (Type A: brute force; Type B: selected search; chess master-like reasoning — later Type C genoemd), evaluatiefuncties met gewichtsfactoren, stability search, de twee dimensionale bordrepresentatie, en een filosofische beschouwing over het spelen van een redelijk briljante partij.

Turing volgde op gepaste afstand wat computerschaak betreft, maar legde wel eerst de grondslag voor de kunstmatige intelligentie met zijn artikel 'Computing machinery and intelligence' dat in 1950 werd gepubliceerd. In 1953 volgde: 'Digital computers applied to games'. Het werd opgenomen in het boek van B.V. Bowden, getiteld *Faster than Thought*. Turing beschreef naast de waardering van de stukken ook de waardering van de positionele kenmerken. Verder onderscheidt hij het begrip 'quiescence' dat bij Shannon onder stability search werd behandeld. Interessant is dat Turing samen met David Champernowne ook handsimulaties uitvoerde. Hun programma had de naam TUROCHAMP.

Hoewel Shannon en Turing eigenlijk zonder discussie de eerste twee grote onderzoekers zijn op het gebied van computerschaak, zijn zij voorafgegaan door de Hongaarse onderzoeker Tihamér Nemes (1895–1960) die soortgelijke ideeën had en ze publiceerde in *Magyar Sakkvilág* en *Chess* in 1949. Later publiceerde hij uitvoerige beschrijvingen in *Acta Technica*. Door de koude oorlog duurde het lang voordat deze informatie tot het Westen doordrong.

De ideeën van Shannon en ook van Turing waren mede gebaseerd op experimenteel-psychologisch onderzoek van A.D. de Groot (1914–2006), die in 1946 promoveerde op het proefschrift *Het Denken van den Schaker*, alsmede op de note van Norbert Wiener (1894–1964) die is opgenomen in zijn boek *Cybernetics, or Control and Communication in the Animal and the Machine*. Wiener denkt dat het mogelijk moet zijn dat een machine een 'goede' partij schaak speelt.

1950–1960 De eerste conferentie over computerschaak

Na de grote onstuimige vloed van ideeën van de kant van Shannon en Turing was het rustig aan het computerschaakfront. Dieter Gunther Prinz wist op een MADM computer aan de universiteit van Manchester het eerste schaakprogramma te implementeren in 1951. Het programma was in staat een mat-in-twee-probleem op te lossen. Om het zoekprogramma te versnellen ontwikkelde Prinz de killer heuristiek. Dat bleek een bijzonder waardevolle bijdrage te zijn voor alle programma's. Omstreeks deze tijd hield Christopher Strachey, een andere talentvolle medewerker van Turing, zich bezig met draughts (dammen op de honderd velden).

Ook in Los Alamos (New Mexico) raakten onderzoekers geïnteresseerd in intelligente systemen die konden schaken. Van Oppenheimer mocht er in de middagpauzes aan dergelijke 'frivoliteiten' worden gewerkt.

Computerbeperkingen leiden tot een beperking van de bordgrootte. Het werd een 6×6-bord (de lopers verdwenen). Dit betekende dat de verhouding van Paard en Toren danig veranderde. De eerste partij werd gewonnen tegen een secretariële medewerkster die speciaal voor deze partij de loop der stukken had geleerd. De tweede partij werd gewonnen door Martin Krushkal (Princeton) die verplicht werd het programma een dame voor te geven. In het verslag van deze twee partijen wordt eveneens Russisch onderzoek door V.M. Kurochin vermeld.

De grote doorbraak in dit decennium was evenwel de Dartmouth-conferentie die in 1956 plaatsvond. John McCarthy was de organisator samen met Marvin Minsky, Nathaniel Rochester en Claude Shannon. Hun opdracht was te onderzoeken of een computer in de toekomst 'intelligentie' en 'leren' zou kunnen simuleren. Onder de deelnemers zien we Herb Simon, Allen Newell, Arthur Samuel en Alex Bernstein.

In 1958 gelukte het de Bernstein-groep (IBM) een schaakprogramma te laten spelen op een 8×8-bord. De aandeelhouders van IBM waren 'not amused' dat hun geld aan dergelijke frivoliteiten werd besteed en T.J. Watson Jr. gelaste dat het onderzoek werd gestopt.

Aan de Amerikaanse universiteiten ging het onderzoek verder. Er zijn drie groepen die de ontdekking van de alfa-beta-algoritme claimen (een zoekalgoritme gebaseerd op de minimax-methode met een uitgekiend snoeimechanisme). Newell, Shaw en Simon waren de eersten die erover publiceerden (1958). Het bleek later de eenvoudige snoeiing te zijn. Hun NSS-programma was gebaseerd op de ideeën van De Groot. Het programma was geschreven in een hogere programmeertaal (ook nieuw in die tijd). Samen met George Baylor implementeerde Herb Simon de *null move* in het programma Mater III om te ontdekken of er sprake was van mat-in-1. Dit is het begin geworden van een krachtig snoeimechanisme in het alfa-beta-zoekproces.

De naam *alfa-beta-algoritme* komt overigens van McCarthy. Hij bedacht het toen hij naar een lezing van Bernstein luisterde in de Dartmouth-conferentie. Hij implementeerde het, naar eigen zeggen, als eerste maar vond het niet nodig om het op te schrijven (de algoritme was vanzelfsprekend). Dit is een grappige constatering, want toen Samuel in 1959 zijn grote standaard werk over AI en checkers (dammen op 64 velden) schreef, besteedde hij om dezelfde reden geen aandacht aan de alfa-beta-algoritme. Dat deed hij pas in 1967. Dat was het eerste standaard werk over de alfa-beta-algoritme. Let wel, de algoritme werd in die tijd nog altijd beschouwd als heuristiek en niet als een 'echt' algoritme. In het artikel uit 1959 legde Samuel de nadruk op het leren van de gewichtsfuncties in de evaluatiefunctie. Dat was een doorbraak in het denken over leren (zie Dartmouth-conferentie). Daarmee heeft Samuel toen de basis gelegd voor de huidige successen in machine learning.

1960–1970 De computer speelt mee in een schaaktoernooi

In dit decennium verschoof het onderzoek van slim zoeken naar het *representeren* van kennis. Vooral John McCarthy was een aanhanger van meer kennis in programma's. Aan het MIT vond hij in 1961 bachelorstudent Alan Kotok bereid om een schaakprogramma van dit type te ontwikkelen. Naast veel aandacht voor de evaluatiefunctie implementeerde Kotok een éénzettengenerator (in plaats van twee). Na zijn studie verdween Kotok uit de computerschaakwereld, maar McCarthy nam Kotoks programma mee naar Stanford en werkte daar verder aan de ontwikkeling van een *plausibele* zettengenerator (Shannon B-strategie).

Intussen was aan het ITEP (Instituut voor Theoretische en Experimentele Physica) in Moskou ook de ontwikkeling van een schaakprogramma doorgegaan, onder leiding van George M. Adelson Velsky. Het programma maakte gebruik van een M-20-computer en was ontwikkeld volgens de Shannon A-strategie (brute force).

Op 22 november 1966 begon een match van vier partijen tussen Stanford en ITEP. De partijen vonden gelijktijdig plaats. De zetten werden doorgegeven per telegraaf. Na veertig zetten werd er afgebroken en werden de stellingen geanalyseerd. De eerste overwinning kwam op 10 maart 1967 tot stand. ITEP zette Stanford mat op zet 19. ITEP won nog een partij. Twee partijen werden afgebroken. In beide partijen stond ITEP beter. De eindstand werd bepaald op 3-1. De uitkomst werd gezien als een succes voor brute force.

Deze historische ontmoeting werd gevolgd door een nieuwe doorbraak onder leiding van Richard D. Greenblatt, evenals Alan Kotok een student aan MIT. Greenblatt begon medio november 1966 aan het bouwen van het schaakprogramma Mac Hack Six. In februari 1967 schreef hij het programma in als deelnemer aan een plaatselijk schaaktoernooi. Het was de eerste keer dat dit gebeurde. Mac Hack Six scoorde 0,5 uit vijf partijen, goed voor een rating van 1243 (Klasse D). In het programma waren vier nieuwe ideeën geïmplementeerd: meer kennis in de evaluatiefunctie, meer kennis in de plausibele zettengenerator, een openingsbibliotheek en de implementatie van *secondary search* (een soort *progressive deepening*, ontleend aan De Groot).

Eigenlijk werd het decennium gekarakteriseerd door Greenblatt en Dreyfus. Greenblatt met zijn doorbraak (zelfs Robert J. Fischer heeft tegen het programma gespeeld) en Dreyfus met zijn oppositie tegen Kunstmatige Intelligentie. Het begon in 1965 met de publicatie van 'Alchemy and Artificial Intelligence' dat de kern werd van zijn later publicatie *What Computers Can't Do; A Critique of Artificial Reason*. Simons voorspelling uit 1957 "Unless the rules bar it from competition, within ten years a computer program will be the world champion" werd in 1965 door Dreyfus op de hak genomen met "still no chess program can play even amateur chess, and the world championship is only 2 years away." In 1967 nodigde Seymour Papert de AI-criticus Dreyfus uit voor een partij tegen Mac Hack Six. Dreyfus verloor in 37 zetten. Het zegt niets, alleen dat Dreyfus niet goed is in schaken.

1970–1980 Het eerste wereldkampioenschap computerschaak

Zoeken en alles willen weten, zou dat te combineren zijn in het schaken? Misschien is het voor het gehele spel iets te hoog gegrepen, maar in eindspelen zou het toch mogelijk moeten zijn om tot perfecte kennis te komen. Zo dacht ook Thomas Ströhlein. Het gaat dan immers om het opslaan van de stellingen met hun bijbehorende theoretische waarde (gewonnen, remise, verloren), gevolgd door een getal dat aangeeft hoeveel zetten die bepaalde positie nog van het eindresultaat verwijderd is. Dit is de meest eenvoudige vorm van een database. In 1970 was Thomas Ströhlein de eerste onderzoeker die deze constructie in de computerschaakwereld bracht en wel voor de eindspelen KTK, KDK, KTKL, KTKP, KDKT (K=Koning, T=Toren, D=Dame, L=Loper,

De stelling uit de partij Timman–Velimirovic (1979). Is deze stelling gewonnen voor wit? Dat was een onderzoeksvraag in het computerschaak in de jaren tachtig van de vorige eeuw.

P=Paard). Daarna volgden veel onderzoekers zijn voorbeeld. Er kwamen nieuwe metrieken, zoals afstand tot mat, afstand tot conversie. De subtiliteit ligt voor een deel in de remiseregel dat binnen vijftig zetten er een stuk geslagen moet zijn of een pion verschoven (afgezien van met name benoemde uitzonderingen). Een beroemd voorbeeld is Timman–Velimirović (1978) dat in 1986 volledig is uitgeanalyseerd met de databasetechniek.

Tot 1975 werd de alfa-beta-algoritme beschouwd als een heuristiek, een goed werkende heuristiek, maar niet meer dan dat. Brudno (1963) was eigenlijk de eerste onderzoeker die getracht heeft een lijn te brengen in de experimentele resultaten tot dan toe. Er waren twee ideeën in omloop: (1) de algoritme is buitengewoon gevoelig voor de volgorde van de te onderzoeken stellingen (vandaar dat eerst schaakzetten, slagzetten en killerzetten onderzocht werden). Toen kwam J.J. Scott in 1969 met het idee om (2) de volgorde van de zetten door de algoritme zelf te laten bepalen. Dit is het idee van de iteratieve alfa-beta-algoritme. Hij is later verscheidene keren onafhankelijk herontdekt, bijvoorbeeld door Jim Gillogly (Carnegie Mellon), David Slate en Larry Atkin (North Western University) en het Russische Kaissa-team.

In een fundamenteel artikel dat in 1975 gepubliceerd werd, tonen Donald Knuth and Ronald Moore aan dat de toepassing van de alfa-beta-algoritme altijd dezelfde waarde oplevert als de minimax-algoritme. Daarmee werd de algoritme van zijn heuristische karakter ontdaan. Voor de theoretici was het een grote stap. De experimentele onderzoekers hadden dit altijd al gedacht.

Natuurlijk was het voor hen (de experimentele onderzoekers) nog steeds van het grootste belang om de beste zet het eerst te kunnen onderzoeken. Kortom, heuristieken zijn nodig voor tijdwinst. Dat geldt ook voor denken in de tijd van de tegenstander. Dus als je een *principal variation* kunt bepalen dan kan je anticiperen op de zet van de tegenstander en op die manier alvast een zet voorbereiden. In het gunstigste geval kun je a tempo antwoorden als de tegenstander zijn/haar zet speelt. Ook in de representatie van bord en stukken vonden veranderingen plaats door toepassing van de bit-representaties (afhankelijk van de woordlengte van de computer). Hieraan zijn de namen verbonden van Adelson Velsky en zijn team (1970). Onafhankelijk is het herontdekt door Hans Berliner (1974) en ook door Slate en Atkin (1977).

Al deze zaken begonnen belangrijk te worden omdat speciale computerschaaktoernooien het licht begonnen te zien. De belangrijkste zijn: het North American Computer Chess Championship (NACCC) dat georganiseerd werd door de ACM en plaatsvond van 1970 tot 1994 en het World Computer Chess Championship (WCCC), dat aanvankelijk georganiseerd werd door IFIP (1974–1977) en daarna door de ICCA/ICGA (vanaf 1980). Eerst eens in de drie jaar, daarna onregelmatig met de intentie ieder jaar een toernooi te organiseren.

Voorts is er van 1980 tot 2001 een World Microcomputer Chess Championship (WMCC) geweest en wordt er sinds 2010 ook een World Chess Software Championship (WCSC) georganiseerd. De resultaten staan op de Wikipedia-pagina over de WCCC. Sinds 2010 is er ook een Unofficial WCCC dat onder de naam van Top Chess Engine Championship (TCEC) wordt georganiseerd.

Voor het Russische Kaissa-team was het winnen van het eerste wereldkampioenschap voor computers in Stockholm 1974 een geweldige onderstreping van hun activiteiten op dit gebied. De belangrijkste bijdragen voor dit succes kwamen van Mikhail Donskoy en Vladimir Arlazarov. Het tweede kampioenschap in Toronto 1977 werd gewonnen door het programma Chess 4.6 van David Slate en Larry Atkin. Aan het einde van dat kampioenschap werd de International Computer Chess Association opgericht onder voorzitterschap van Benjamin Mittman die ook hoofdredacteur werd van de *ICCA Newsletter*.

Een jaar na het eerste wereldkampioenschap werd in 1975 de eerste Advances in Computer Chess Conference (ACC) georganiseerd in Londen. Dit was het begin van een zeer vruchtbare uitwisseling van ideeën over computerschaaktechnieken. Aanvankelijk werden ze georganiseerd met een interval van drie jaar; later werd dit om de twee jaar met daartussen in de serie Conferences on Computers and Games (CG). Voor een volledige opsomming van boeken en *proceedings* verwijzen we naar de overvloedige literatuur.

Door deze bijeenkomsten werd de uitwisseling van informatie aanzienlijk versneld en vond er een grotere integratie van AI-technieken en computerschaaktoepassingen plaats. Met name het werk van Arthur Samuel vond zijn weg. Zo riep het werk van J.R. Quinlan (1979), 'Discovering rules by induction from large collections of examples', gepubliceerd in D. Michie (ed.), *Expert Systems in the Micro Electronic Age*, Edinburgh University Press, veel positieve reacties op van onderzoekers die begrijpelijke (*human intelligible*) regels wilden afleiden uit eindspeldatabases (zie Ströhlein, 1970).

1980–1990 Ken Thompson bouwt Belle

In 1980 werd de kennisexplosie zichtbaar in erkenning, verdere groei van activiteiten en een groot aantal publicaties van nieuwe ideeën. Veel van deze ideeën vonden hun weg in de computerprogramma's van de onderzoekers. In dit decennium valt de nadruk op parallellisme. De toponderzoeker uit die tijd is zonder enige twijfel Kenneth Lane Thompson (1943). In de periode 1969–1985 verrijkte hij de wereld met bijdragen aan het besturingssysteem Unix (samen met Dennis Ritchie). Hij ontwierp de taal B en was betrokken bij de ontwikkeling van de taal C. Samen met Joe Condon bouwde hij het schaaksysteem Belle (het was niet alleen een programma maar een technologische entiteit). Belle was het eerste programma dat in 1983 officieel de US Chess Master titel ontving en op grond daarvan de eerste Intermediate Fredkin Prize van $5000 kreeg toegekend. Verder genereerde Thompson ook veel nieuwe vijf-man-eindspeldatabases. Belle bracht hem in Linz 1980 de wereldtitel. In 1983 ontving hij samen met Dennis Ritchie tijdens het wereldkampioenschap computer-

schaak in New York de Turing Award voor hun hierboven genoemde bijdragen en vervolgens in 1998 de National Medal of Technology uit handen van Bill Clinton. Het was een grote verrassing dat hij in 1983 Belles wereldtitel niet prolongeerde, maar werd opgevolgd door Cray Blitz. Dit maakte toen ineens voor iedereen duidelijk: computerschaak is een combinatie van hardware en software. Robert Hyatt, Harry Nelson en Albert Gower hadden snelle transpositietabellen en nog snellere multiprocessors. De Cray Supercomputer was geboren en in gebruik genomen.

De ideeënontwikkeling ging vervolgens in een razend tempo verder. Het ene idee was nog niet gepubliceerd of het ander was alweer gelanceerd. In 1980 publiceerde Judea Pearl zijn Scout-algoritme en dat was een schot in de roos, want goede ideeën zijn niet zelden de aanleiding voor nog betere ideeën. In 1983 had ik als hoofdredacteur de *ICCA Newsletter* omgevormd tot *ICCA Journal*. De ontvangst was fantastisch. In het eerste nummer ontving ik een bijdrage van Botwinnik. Daarna volgde Jonathan Schaeffers History Heuristic. Voor het tweede nummer stuurde Alexander Reinefeld mij het artikel 'An improvement to the scout tree search algorithm'. Hij noemde het Negascout. Het werd een succes in menig schaakprogramma. In 1987 introduceerde Don Beal Null Move Quiescence Search (NMQS) om ook in de uiteinden van de zoekboom zo efficiënt mogelijk te werk te gaan.

Organisatorisch gezien gebeurde er eveneens veel, ook in Nederland. Op 18 oktober 1980 werd de Computer Schaak Vereniging Nederland (CSVN) opgericht. Er waren direct al 622 leden. De CSVN zou vervolgens ieder jaar een Nederlands kampioenschap houden. International besloot de ICCA om vanaf 1980 ieder jaar een World Microcomputer Chess Championship te organiseren. Dat gebeurde zo tot 2001.

Intussen was de TU Delft ook in de ban van het computerschaak geraakt. Dat kwam door mijn eigen onderzoek: promotie in 1983 op het proefschrift *Computerschaak, Schaakwereld en Kunstmatige Intelligentie* (promotors H.J.M. Lombaers, A.D. de Groot en S.J. Doorman). Bovendien vond de studievereniging Christiaan Huygens het ook een mooi onderwerp voor haar vijfde lustrum in 1982. Een openbaar debat in 1981 op de achterkant van *NRC Handelsblad* tussen Grootmeester Donner en mij leidde op 5 maart 1982 tot de ontmoeting Donner–Belle. Het was een fascinerende bijeenkomst, niet in het minst door de aandacht van de pers. Donner won in 56 zetten, maar de trend was gezet. Donner vergeleek het spel van Belle met dat van zijn nichtje van vier, maar ik zag het als een stap voorwaarts in de richting van een match met de wereldkampioen. Naast het programma Pion (Derksen en Huisman) legde het Delftse onderzoeksteam (Dekker, Nakad, Van den Herik) zich ook toe op het creëren van bijzondere vijf- en zes-man-eindspeldatabases, zoals twee paarden tegen een pion en het Timman–Velimirović-eindspel (toren tegen loper en aan beide zijden een a-pion). Harry Nefkens ontwikkelde de openingsbibliotheek voor Pion en koos voor het opslaan van stellingen in plaats van varianten. Zo kon hij vele grootmeesterpartijen aan de openingsbibliotheek toevoegen. Dit inspireerde Dap Hartmann tot het bouwen van de zogeheten Dap Tap, een mechanisme om patronen te herkennen die het zoekproces kunnen ondersteunen. Deze techniek is later ook in Deep Blue gebruikt. In die tijd publiceerde De Groot 'Intuition in chess' (1986). Hij dacht dat intuïtie een essentieel onderdeel was om schaakgrootmeester/wereldkampioen te kunnen zijn. Volgens hem viel intuïtie niet te programmeren. Dus een computer zou de wereldkampioen nooit kunnen verslaan.

Ken Thompson, Claude Shannon en David Slate bij het 6e wereldkampioenschap computerschaak in Edmonton (Alberta), 1989. Bron: Monroe Newton.

In diezelfde tijd was er aan Carnegie Mellon University grote activiteit te bespeuren. Daar werkte sinds 1969 Hans Berliner (1929–2017), de vijfde wereldkampioen correspondentieschaak (1965–1968), aan computerschaak-onderwerpen. Hij promoveerde in 1974 bij Allen Newell op *Chess as Problem Solving: The Development of a Tactics Analyser*. Hij kreeg onvoldoende greep op de evaluatiefunctie en besloot nieuwe ideeën toe te passen op Backgammon. Het resultaat was dat zijn programma BKG9 (gebaseerd op fuzzy logic) in juli 1979 wereldkampioen Luigi Villa versloeg met 7-1. Het was voor het eerst dat een wereldkampioen in een willekeurige sport van een computerprogramma verloor. Vervolgens ontwikkelde Berliner de B*-algoritme. Deze en andere ideeën lagen ten grondslag aan HiTech. Met zijn team (onder anderen Carl Ebeling en Murray Campbell) werkte hij dag en nacht aan het nieuwe programma, dat gebaseerd was op multiprocessing, speciale circuits en kennis-uitbreidingen via transities.

Een belangrijke bijdrage in die tijd was ook 'Using time wisely' van de hand van Bob Hyatt (1984). Hiermee gaf hij de lezer een goede indruk hoe de tijd verdeeld moest worden over de verschillende taken die zich binnen een programma afspelen. Voorts is zeer noemenswaardig Hash Tables in Cray Blitz (Nelson, 1985) en de ontwikkeling van nieuwe Search Tables (Warnock en Wendroff, 1986). Daarnaast werden nieuwe zoektechnieken ontwikkeld zoals *Parallel alpha-beta search* (Newborn,

Jan Hein Donner versus Belle

In 1982 werd Jan Hein Donner uitgedaagd tot een partij tegen Belle, door de auteur van dit artikel, via een open brief in *NRC Handelsblad*. In die tijd dachten schakers dat de computer nooit zou winnen van de mens. Dat blijkt ook wel uit het volgende fragment van het antwoord van Donner in dezelfde krant:

> "Ik weet heel goed wat computers kunnen of hoe een schaakprogramma opgezet zou kunnen worden, maar juist daardoor weet ik ook hoe waardeloos de dure prullen zijn die jij regelmatig in *De Telegraaf* aanbeveelt. Het vraagstuk van de zogenaamde Artificiële Intelligentie voert zeker tot boeiende gedachtenexperimenten, maar waar ik bezwaar tegen maak, is dat dit begrip door de elektronische industrie gebruikt wordt om mensen knollen voor citroenen te verkopen... Overigens wil ik best tegen jullie 'wereldkampioen' een schaakje zetten, hoe belachelijk dat ook moge zijn... Helaas leert de ervaring dat je al reclame voor die onzin maakt, door er uit de losse pols tegen te spelen. 'Speelde tegen een echte grootmeester' zeggen jullie dan, of 'Donner had er toch maar 39 zetten voor nodig'. Dat is stuitend, maar goed, ik ben nu eenmaal beroepsspeler."

Uiteindelijk kwam het tot een partij op 5 maart 1982, gespeeld op de Faculteit Wiskunde en Informatica van de TU Delft. Donner won overtuigend, in 56 zetten.

1982; Bal en Van Renesse, 1989) en Conspiracy Number Search (McAllester, 1987).

Na tweemaal een wereldkampioenschap van Cray Blitz (1983 en 1986) wist Deep Thought (net als HiTech afkomstig van Carnegie Mellon) in 1989 de wereldtitel te veroveren. De strijd tussen de twee Carnegie Mellon-teams was adembenemend, te meer omdat Murray Campbell voor beide teams werkzaam was. Hij was de *trait d'union* die alle geheimen bij zich droeg. In de eindstrijd zat hij samen met Hans Berliner achter de stukken van HiTech tegenover Feng-hsiung Hsu en Thomas Anantharaman, die de zetten voor Deep Thought uitvoerden.

Toch is het goed om even bij deze interne competitie stil te staan. Hans deed het goed, hij was een *crack*. Maar Feng-hsiung Hsu was een *rising star* die wilde doorbreken. Hij wist alles beter en dat was ook zo. Hij was intelligent, ontwierp in 1985 Chiptest (in 1988 Deep Thought genoemd) en verzamelde slimme mensen om zich heen. Hans aasde op de Fredkin Intermediate Prize van $10.000. Inderdaad versloeg HiTech Grootmeester Denker (die in Groningen 1946 remise tegen Botwinnik had gespeeld) in 1989, maar een jaar eerder had Deep Thought in een sterk grootmeestertoernooi al van de Deense Grootmeester Bent Larsen een echte serieuze partij gewonnen. Het was de eerste keer dat een grootmeester van wereldklasse door een programma verslagen werd. Daarom werd in 1988 de tweede Intermediate Fredkin Prize aan Deep Thought toegekend.

1990–2000 Deep Blue verslaat de wereldkampioen

Alle bovengenoemde activiteiten waren in het laatste decennium van de vorige eeuw mogelijke bronnen van inspiratie tot grote successen voor twee Nederlandse programmeurs. Ze waren geen academische onderzoekers, maar ze waren vakmensen van het hoogste niveau, van wereldklasse. In Vancouver 1991 won Ed Schröder met het programma Gideon het WMCC, zij het gedeeld met Mephisto. In Madrid 1992 veroverde Ed Schröder daarenboven het wereldkampioenschap (WCCC) in alle categorieën met het programma The ChessMachine Schröder. Het was een microcomputer, maar met een formidabele sterkte. Schröder had zich in Keulen 1986 met het programma Rebel al eerder in de kijker gespeeld door tot in de laatste ronde kanshebber te zijn voor de titel. Nu, in 1992 was het dan zover. Het leverde hem veel roem en erkenning op.

Na 1989 had het Deep Thought team zijn zinnen gezet op het ontwikkelen van een programma dat de menselijke wereldkampioen zou verslaan. Daarbij werden ze gesteund door IBM. Als onderzoekers en bouwers was het team heel hard bezig met het bereiken van hun doel en daarom had het zich teruggetrokken van de gewone competities. Na zes jaar kwam het team tevoorschijn bij het WCCC 1995 in Hong Kong en speelde daar onder de naam Deep Blue. Het toernooi was bijzonder sterk bezet. Vanuit MIT speelde Star Socrates mee, een programma van Don Dailey, Christopher Joerg en Charles Leiserson, met schaaktechnische ondersteuning van Larry Kaufman. Zij werden uiteindelijk tweede. Om een lang verhaal kort te maken, in de laatste ronde speelde het programma Fritz van de andere Nederlandse top-programmeur Frans Morsch (toen 3 uit 4) tegen Deep Blue (toen 3,5 uit 4). In een heel spannende partij won Fritz overtuigend, zonder Deep Blue ook maar een enkele kans te geven. Fritz was wereldkampioen. Star Socrates was tweede en Deep Blue derde. Wat nu?

Aan het einde van het toernooi besloot IBM zijn plannen toch te onthullen en was er een persconferentie waarop meegedeeld werd dat IBM met Deep Blue een poging ging wagen om in 1996 wereldkampioen Kasparov uit te dagen met een verbeterde versie van het huidige programma. Dat was de inleiding tot een nieuwe fase in de historie van het computerschaak (Zuses droom was nu wel heel dichtbij gekomen).

In 1994 lanceerde Bob Hyatt het opensourceprogramma Crafty. Tot dan toe was er slechts een redelijk sterk opensourceschaakprogramma genaamd GNU Chess, het was een loot van het bekende GNU-project. Nu was er een programma dat kon dienen als tegenstander, oefenpartner, en zelfs als deelnemer aan de wereldkampioenschappen (een door een Indonesisch team verbeterde versie speelde in Jakarta 1996 mee voor het wereldkampioenschap met instemming van de deelnemers). Hyatt is een pionier die veel heeft bijgedragen aan de ontwikkeling (zie eerder). Hier noemen we: een evaluation cache, transpositietabellen, bit-board datastructures, en rotated bitmaps.

Al dit onderzoek was mede-gestimuleerd door het initiatief van David Levy om in 1989 samen met Don Beal een computerolympiade te organiseren in Londen. Van de opeenvolgende computerolympiades leerden de onderzoekers twee zaken: (1) computerschaak was nog altijd de stimulans om andere spelen te onderzoeken, en (2) misschien was het moeilijkste spel ter wereld, Go, wel een waardige onderzoeksopvolger van het edele schaakspel. Duidelijk was dat er verschuiving van aandacht plaatsvond. Dit leidde ertoe dat de ICCA in 2002 officieel van naam veranderde van

ICCA naar ICGA (International Computer Games Association).

De grootste omwenteling in het decennium 1990–2000 was het verslaan van Kasparov. Het werd voorafgegaan door een oefening in wachten en verbeteren. In de eerste match Kasparov–Deep Blue (1996) speelde Deep Blue op een RS/6000 SP-machine met 36 processoren. De eerste partij werd op 11 februari 1996 gespeeld en leek voorbestemd om een nieuw tijdperk in te luiden, omdat Deep Blue overtuigend won. Dat bleek evenwel niet het geval te zijn, want Kasparov besliste de match met 4–2 in zijn voordeel. Een jaar later (1997) kwam er een tweede match. De hardware en de structuur van het programma bleven nagenoeg hetzelfde. Er werd meer getraind met grootmeesters (Larry Christiansen en Michael Rohde), hoewel ook voor de eerste match drie grootmeesters testpartijen hadden gespeeld.

De tweede match begon met een enorme dreun voor het IBM-team, Kasparov won overtuigend. Het Deep Blue-team likte de wonden en vatte nieuwe moed. In de tweede partij liet Deep Blue sterk openingsspel (gesloten Spaans) zien en bracht Kasparov in een moeilijke positie. Als Deep Blue toen op pionwinst gespeeld had, zou de partij tactisch zeer gecompliceerd zijn geworden. Deep Blue speelde echter de prachtige zet 37. Le4. In hogere zin was de partij beslist. Weliswaar speelde Deep Blue daarna met 44. Kf1 nog een zwakke zet, maar uiteindelijk won het programma en daarmee kwam de stand op 1-1. Het werd vervolgens 2,5-2,5. De match zou dus in de zesde partij beslist worden. Kasparov speelde te uitdagend in de opening en kwam via een stukoffer in een verloren positie terecht. Een verdiende overwinning van Deep Blue. Zuses droom was vervuld. De verkoop van de IBM-computers steeg, maar Kasparov speelde zijn laatste kaart. Hij verklaarde op persconferenties, in publicaties en op gewone conferenties dat grootmeesterlijke (dus menselijke) hulp ten grondslag lag aan 37. Le4. Zo was ook 44. Kf1 een menselijke fout. Beide zetten zouden nooit door een weldenkende computer gespeeld kunnen zijn. De eerste zet was te goed en de tweede te slecht. Voor zijn prestatie ontving het Deep Blue-team de Fredkin Prize van $100.000. IBM heeft op grond van publicitaire overwegingen Deep Blue niet meer laten spelen. Einde van een prachtig traject. Toch gingen de ontwikkelingen verder.

2000–2010 Schakers zijn kansloos tegen de computer, wanneer volgt Go?

De grote vraag van het eerste decennium in de nieuwe eeuw was: wanneer kunnen microcomputers de menselijke wereldkampioen verslaan? Een tweede gerelateerde vraag was: wanneer gaan grote (of micro-) computers de ELO-grens van 3000 punten passeren? Tijdens de WCCC 1999 was het de microcomputer Shredder gelukt om in een beslissingspartij om de wereldtitel het programma CilkChess (spelend op een supercomputer) te verslaan. Hardware en snelheid zijn belangrijk, maar slimme algoritmen spelen hun eigen rol. In 2001 werd de laatste WMCC gespeeld. In 2003 speelde Kasparov tegen veelvuldig WMCC-winnaar Deep Junior een match van zes partijen. Het werd 3-3. In 2005 speelde Hydra tegen Michael Adams (ooit nummer 4 op de wereldranglijst): de uitslag was 5,5–0,5. Daarna speelde Deep Fritz tegen toenmalig wereldkampioen Kramnik een match van zes partijen. Deep Fritz won afgetekend met 4-2. Daarmee was de eerste vraag beantwoord. Ook een microcomputer was sterker dan de wereldkampioen.

Door de combinatie van WCCC, olympiade en conferentie was er veel interactie en zag het Go-onderzoek kans om verder op te rukken in de belangstelling. Aanvankelijk ging het traag omdat de alfa-beta-minimaxmethode niet erg geschikt was voor Go (het lag niet aan de branching factor, de grote zoekruimte of de complexiteit zoals menigeen dacht, maar aan de moeilijkheid van de juiste evaluatie — zie ook de problemen die Berliner had in 1974–1979). De Franse onderzoeker Bruno Bouzy had een oplossing: vermijd alle evaluaties en kijk tot het einde van de variant en doe dat voor een grote (beperkte) verzameling van zetten die random gekozen zijn.

De implementatie van het idee in het programma Indigo leidde in de olympiadetoernooien tot posities in de middenmoot, tot publicaties in de *ICGA Journal* en tot zijn Habilitation thesis (2004). Het bleek na verloop van tijd evenwel de opening naar een nieuw terrein van onderzoek. Op de CG 2006 conferentie lanceerde Remy Coulom de gedachte om met Monte Carlo-searchtechnieken de zoekruimte te doorkruisen. Iedereen in de zaal in Turijn voelde: dit is de richting die we voor Go moeten gaan. Twee belangrijke bijdragen aan de verdere ontwikkeling van deze technieken zijn van (1) Kocsis en Szepesvári (2005) die de UCT-formule ontwierpen en (2) Chaslot, Winands, Uiterwijk, Van den Herik en Bouzy (2008) die de Monte Carlo Tree Search (MCTS) ontwierpen, waarmee de ideeën van Coulom in een dedicated framework werden ondergebracht. In deze zich onstuimig ontwikkelende Go-wereld ging het computerschaak zijn eigen weg: naar perfectie van speelsterkte en perfectie van efficiënte algoritmen.

In 1999 had Shredder de hegemonie van de supercomputer gebroken. Vanaf dat moment zien we dat de microcomputers Shredder en Deep Junior de WCCC domineren. In 2005 kwam daar verandering in. Twee programma's kwamen uit het niets en regeerden met straffe hand de WCCC in Reykjavik. Zappa (Anthony Cozzie) werd wereldkampioen met 10,5 uit 11 en Fruit (met Fabien Letouzey) werd tweede met 8,5 uit 11. Het leek even dat er een nieuwe tijd aanbrak maar het liep anders. Zappa verdween na 2007 uit de arena en Fruit had zijn programma publiekelijk beschikbaar gesteld om tegen te spelen en ideeën uit op te doen, maar niet om daar modules aan te ontlenen. Zelf ging Letouzey iets anders doen om vervolgens jaren later mee te spelen in de computerolympiade met het Draughts-programma Scan. Junior (zonder Deep) en Shredder hernamen hun vroegere posities en in Turijn 2006 werd Junior wereldkampioen; Shredder en Rajlich (later Rybka genoemd) werden gedeeld tweede. In Amsterdam 2007 toonde Rybka zich de sterkste. Dat bleef zo in Beijing 2008, Pamplona 2009 en Kanazawa 2010.

In de Go-wereld was een soortgelijke trend waarneembaar. Daar trok het programma MoGo alle aandacht naar zich toe. De kracht van MCTS nam steeds meer toe, omdat de juiste wijze van kennis toevoegen aan het zoekproces was gevonden; nog wel niet volledig, maar toch in voldoende mate om het random effect te sturen. Volgens sommigen zou het komende decennium het sluitstuk vormen, volgens anderen lag dat zeker nog wel een decennium verder.

2010–2016 De strijd tussen de schaakcomputers

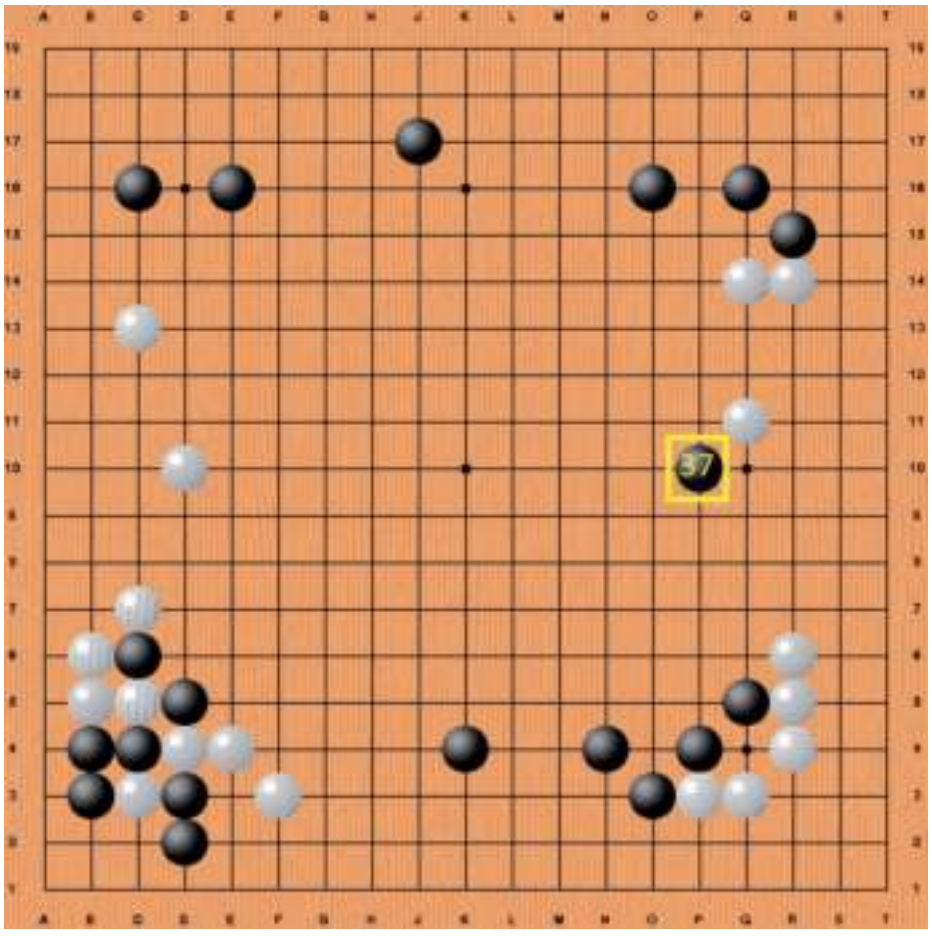
Schaken bleef in de belangstelling en computerschaak helemaal, ook nadat de menselijke wereldkampioen was versla-

gen. Veel techniekgeïnteresseerden maakten hun eigen programma's uit andere programma's. Het leidde tot supersterke programma's zoals Houdini en Stockfish. Alleen vond de ICGA dat ze niet aan het wereldkampioenschap konden deelnemen, omdat het daar ging om originele programma's, dat wil zeggen het deelnemende programma moest door de programmeur (of het team van ontwikkelaars) bedacht en geïmplementeerd zijn. De regels voor toelating werden strenger en af en toe werd er een programma geweigerd of zelfs gediskwalificeerd wegens plagiaat. Het was een ingewikkelde clash van wetenschap en techniek, van amateurs en professionals, van uitvinders en dieven. De regels werden gecompliceerder en de druk van de professionele liefhebber (het doet er niet toe waar de algoritmen vandaan komen) werd groter.

Op 23 januari 2011 kwam de zaak Rybka–Fruit tot een openbare clash. Fabian Letouzy schreef een open brief aan het ICGA-bestuur met de mededeling dat de programmeur van Rybka (Vasik Rajlich) zijn werk geplagieerd had. Dat leidde tot diepgaand onderzoek en een technische commissie van het hoogste kaliber (onder anderen Ken Thompson). De commissie deed zijn werk zeer nauwkeurig en kwam tot de conclusie dat de klacht van Letouzy toegewezen diende te worden (er waren zelfs stukken code aangetroffen met oorspronkelijk commentaar van Letouzy). Dit betekende dat Rybka van alle podiumplaatsen in de WCCCs en de WCSCs vervallen werd verklaard. Er kwamen nieuwe wereldkampioenen en nieuwe runner-ups. De computerschaakwereld stond op zijn kop. De ontwikkelingen gingen evenwel snel door. Johannes Zwanzger voegde zich na jarenlang hard werken met het programma Jonny aan de top. Shredder handhaafde zich en Junior verdween van het toneel. Betrekkelijke nieuwkomer was Komodo. Het was een programma uit de MIT-stal van Don Dailey, dat onderhouden en vernieuwd was door Mark Leffler en van nieuwe schaak-ideeën was voorzien door Larry Kaufman. Zij domineerden de jaren 2015–2017, getuige de uitslagen van de WCCCs en WCSCs in Leiden.

2017 De computer verslaat de wereldkampioen Go

Intussen werd de schaakwereld geholpen door de Go-wereld. Het bedrijf DeepMind bleek in staat om met behulp van deep learning een neuraal netwerk te creëren, dat, door tegen zichzelf te spelen, de details van het spel op een tot dan toe ongeëvenaarde wijze in kaart wist te brengen. DeepMind ontwikkelt het programma AlphaGo, dat in 2015 de Europese kampioen Fan Hui verslaat met 5-0. Een verbeterde versie verslaat in 2016 de veelvuldige wereldkampioen Lee Sedol met 4-1. In mei 2017 verslaat het wereldkampioen Ke Jie, algemeen gezien als de beste speler ter wereld, met 3-0.



Zet 37 van AlphaGo in de tweede matchpartij tegen Lee Sedol. De commentatoren reageerden met ongeloof en deden de zet af als een beginnersfout, omdat de steen pardoes in het open veld werd geplaatst. Zoiets zou een mens nooit doen. Lee Sedol was zichtbaar verbaasd, verzonk in gedachten en ontdekte de kracht van de zet. De mens kan dus ook leren van de computer!

In het najaar van 2017 komt DeepMind met machines die zichzelf hebben leren spelen, zonder te zijn gevoerd met menselijke partijen: de Go-machine AlphaGo Zero en de schaakmachine AlphaZero. Na verloop van tijd besluit DeepMind dat de programma's sterk genoeg zijn om ze te testen tegen andere computers. AlphaGo Zero verslaat AlphaGo met 100-0. Als tegenstander van AlphaZero wordt gekozen voor het programma Stockfish, met een geschatte rating van boven 3400. Ter vergelijking, de menselijke wereldkampioen Magnus Carlsen heeft een rating van minder dan 2900. Dat betekent dat, naar verwachting, Carlsen in tien partijen tegen Stockfish er eentje remise houdt en de rest verleist. AlphaZero wint een match van 100 partijen met 28-0 (bij 72 remises), in schaaktermen met 64-36. Einde verhaal. De nabeschouwing laat ik aan de lezer over. Er is nog één onderzoeksuitdaging over: het vaststellen van de speltheoretische waarde van het schaken.

Dankwoord

Graag bedank ik alle mensen met wie ik in de laatste veertig jaar mocht samenwerken. Zonder hen was dit artikel niet mogelijk geweest. Ik zie af van het noemen van namen evenals van het geven van referenties. Een goede lijst zou langer zijn dan het artikel zelf. Wel bedank ik met name Robbert Fokkink, hoofdredacteur, zonder wiens vasthoudendheid dit historisch overzicht niet geschreven zou zijn. Ten slotte bedank ik het redactieteam dat mij heeft ondersteund samen met Ron de Goeij.

Marijn J. H. Heule
Department of Computer Science
University of Texas at Austin, USA
marijn@cs.utexas.edu

Trip to the Proof

Brute trust

Marijn Heule describes the developments in automated reasoning that led to his solutions of the Boolean Pythagorean Triples problem and the Schur number five problem and to improvements in computing the chromatic number of the plane.

An important trade-off in automated reasoning is efficiency versus correctness. Research and development of fully automatic tools focus primarily on performance, while the interactive theorem proving community deeply cares about the trusted core of their tools. Interactive tools have been successful in constructing a formal proof of famous problems, such as the four color theorem. Fully automatic tools, which are frequently used in industry to find bugs in hardware or software, have become significantly more powerful in the last two decades, thereby allowing to solve long-standing open problems. However, their effectiveness also raised questions whether we can trust these results as computer-generated solutions typically cannot be understood by humans.

My roots lie in highly automated tools for propositional logic, known as satisfiability (SAT) solvers. These solvers determine whether there exists a satisfying assignment (or, equivalently, a solution) for a propositional formula. My interest in correctness originates from my post-doc with Armin Biere at the Johannes Kepler University in Linz, Austria. His solvers have been among the strongest and most reliable ones in the community for over a decade. However, we worried about the correctness of one of the techniques in his top solver. This technique, called blocked clause addition, can remove solutions while ensuring that at least one solution remains (if the initial formula has one). Armin considered the implementation of the technique 'experimentally correct' as he tested the solver on a million small problems without encountering a discrepancy. However, after a few weeks of studying the code, I was able to manually produce a formula with a solution, while his implementation of blocked clause addition would claim there is none.

The bug turned out to be a deep conceptual error. In their quest to further improve performance, developers of state-of-the-art SAT solvers have been adding techniques that go beyond the classical proof system for propositional logic, known as resolution. Examples of such techniques are symmetry breaking, Gaussian elimination, and the earlier mentioned blocked clause addition. The ability to remove solutions makes validation challenging: instead of checking whether no solution is removed (as in resolution), one needs to check whether not all remaining solutions are removed.

In the following years I have been working with various colleagues on new proof systems for propositional logic that allow compact expression of all techniques used in state-of-the-art solvers — including those that cannot be succinctly expressed using resolution. These proof systems reason about the *absence* of facts. We call them *interference-based proof systems*, as learning one fact may block learning another one. In contrast, most proof systems for propositional logic, including resolution, reason about the *presence* of facts.

The design goal of the new proof system was to have a single redundancy criterion that covers all techniques and is computable in polynomial time. This does not imply that checking is cheap since the criterion is more complex and general compared to most reasoning techniques used in SAT solvers. However, we were able to implement a checker that is reasonably efficient. The SAT community started to use the proof system in various settings. For example, participants in the international SAT competitions are required since 2013 to certify that a claim that a problem has no solutions. Such a claim is only considered correct if the certificate, known as a proof of unsatisfiability, can be checked. Also, when Boris Konev and Alexei Lisitsa solved the Erdős discrepancy problem using SAT, they produced and checked a proof of unsatisfiability of their result.

In 2015 I wanted to show that proofs of unsatisfiability are a viable option to show correctness of SAT solving results no matter how many computational resources were required to solve the problem. Initially, I looked at the Schur number five problem. This problem dates back to the early 20th century, when Issai Schur asked whether coloring the positive integers with a finite number of colors would result in a monochromatic solution of the equation $a+b=c$. Schur proved in 1916 that this is indeed the case. Schur number $S(k)$ denotes the largest positive integer such that the numbers $[1,S(k)]$ can be colored with k colors such that there is no monochromatic solution of the equation $a+b=c$. Early on, the first three Schur numbers were determined. However, it took five decades to compute that $S(4)=65$ (by Leonard Baumert in 1965).

Victor Marek from the University of Kentucky has been encouraging me to compute $S(5)$ since the day we met at a workshop in Baltimore back in 2008. Over the years I have been trying to compute this number, but it appeared too hard. Victor suggested to tackle a related challenge: Will any coloring of the positive integers

with two colors result in a monochromatic solution of the equation $a^2+b^2=c^2$? In 1980, Ronald Graham offered an award of $\$100$ for the first person to solve this *Boolean Pythagorean Triples problem*. We teamed up with Oliver Kullmann from Swansea University and determined that the numbers $[1,7824]$ can be colored with two colors while avoiding a monochromatic solution of $a^2+b^2=c^2$, while this is impossible for $[1,7825]$.

The paper about the solution of the Boolean Pythagorean Triples problem was mostly a demonstration that

1. SAT solvers are now able to solve hard problems by linear time speedups even when using thousands of cores; and
2. that we can produce a proof of such hard problems that can be validated by third parties.

Quite unexpectedly, we were contacted by an editor of *Nature* regarding the solution and the proof. In the interview I tried to convince her of the importance of the main contributions. However, she appeared only interested (and worried about) the size of the proof: 200 terabytes. Her article on the 'Largest Math Proof Ever' focussed on the lack of understanding of the computer-generated solution. Moreover, according to the article, the clever algorithms were only 'ticking off possibilities'.

Yet there is no such thing as bad publicity. The aftermath of solving the Boolean Pythagorean Triples problem and the article in *Nature* was very positive. On a personal note, I had the honor of meeting several great mathematicians, including Ronald Graham (who gave me the $\$100$ check), Alfred Hales, Timothy Gowers and Tom Hales. Professionally, it was great that the interactive theorem proving community significantly improved the trust story by developing formally verified checkers of proofs of unsatisfiability. There are now verified checkers in three main theorem provers: ACL2, Coq, and Isabelle.

Meanwhile, I was finally able to solve Schur number five: $S(5)=160$. The Texas Advanced Computing Center made this possible by allowing me to use 2400 CPUs for weeks on end. The size of the resulting proof of unsatisfiability was 2 petabytes, roughly ten times larger than the proof of the Boolean Pythagorean Triples problem. The use of the new proof system was crucial as it allowed to compactly express symmetry breaking. Without it, the proof would have been $120\,(=5!)$ times larger. That would have made it impossible to check the proof, even with the vast number of resources at my disposal. The existence of efficient, formally verified proof checkers also raised the bar for validation. The proof was eventually verified using the ACL2 checker, which required 35 CPU years. Schur number five is arguably the hardest problem ever solved using SAT solvers. We can be confident that this immense proof is correct since it can and has been checked using highly trustworthy systems by independent parties.

Photo: Arjen van Lith

Marijn Heule, with in the background an illustration of the Boolean Pythagorean Triples problem

Recently, an unexpected application for proof checking tools emerged: computing the chromatic number of the plane. This problem asks how many colors are required to color all points of the plane such that no two points at distance 1 from each other have the same color. Early on, two elegant proofs were found to show that the number of colors is at least 4 and at most 7. However, hardly any progress has been made since the 1950's. A breakthrough was announced in April 2018: Aubrey de Grey found a 1581-vertex unit-distance graph with chromatic number 5, thereby improving the lower bound. A Polymath project was started to find a smaller graph with this property. Proof checking tools turned out to be useful here. To validate that a graph has chromatic number 5, one needs to show that there exist no valid 4-coloring. SAT solvers are arguably the fastest method to achieve this. The proof of unsatisfiability showing that no 4-coloring exists can be optimized by proof checking tools. From the optimized proof one can extract a subgraph that has chromatic number 5. This method is more successful than randomly dropping vertices that do not lower the chromatic number. The proof checking tools allowed me to find a unit-distance graph with chromatic number 5 that has only 553 vertices.

Although a graph with 553 vertices is still hard to comprehend for humans, this reduction is substantial and represents a step towards understanding why coloring the plane requires at least 5 colors. In fact, the techniques based on optimizing proofs of unsatisfiability may eventually produce the most elegant argument. This would be an interesting twist in the discussion on the usefulness of mechanized mathematics as computers might be able to give the shortest and clearest proofs of theorems. Some computer-generated proofs may be large, because there exists no short proof.

In the coming years, automated reasoning techniques are poised to solve hard problems that have been open for many decades. Mathematical challenges that may be feasible for an automated approach are Ramsey number five, the Collatz conjecture and the chromatic number of the plane. The proofs may reveal crucial insights that might otherwise be overlooked by mathematicians. However, even if the proofs do not provide any understanding, we can be confident that they are correct, as we have highly trustworthy systems that can validate them.

Wim Caspers
Lyceum Ypenburg, Den Haag, en
Faculteit EWI en Lerarenopleiding, TU Delft
w.t.m.caspers@tudelft.nl

Mark Timmer
Twents Carmel College, Oldenzaal, en
ELAN, Vakgroep Docentontwikkeling, UT
m.timmer@utwente.nl

Onderwijs Bespreking eindexamens nieuwe stijl

Op zoek naar denkactiviteiten

Het huidige wiskundecurriculum voor havo en vwo startte per september 2015 in de vierde klassen [1]. Vorig jaar werd de eerste lichting havo-leerlingen geconfronteerd met een eindexamen over dit nieuwe programma, en enkele maanden geleden was het vwo aan de beurt. Wim Caspers en Mark Timmer, beiden vakdidacticus en docent wiskunde, bekijken wat het nieuwe curriculum dit examenjaar heeft opgeleverd, met veel aandacht voor het eindexamen wiskunde B op het vwo.

Met de invoering van de nieuwe examenprogramma's hebben alle wiskundevakken van de havo en het vwo aanzienlijke wijzigingen doorgemaakt. Afgelopen schooljaar maakten voor het eerst op grote schaal de eindexamenkandidaten van het vwo een examen nieuwe stijl. Op kleinere schaal werd er al jaren op pilotscholen geëxperimenteerd met het nieuwe programma en er werd ook al getoetst door middel van pilotexamens, die gewoon voor iedereen toegankelijk zijn op www.examenblad.nl. Op die manier kregen zowel leerlingen als docenten een indicatie van wat er van de nieuwe examens in 2018 verwacht kon worden. Voor de havo was het examen in 2017 al het eerste grootschalige examen nieuwe stijl. We illustreren per schoolvak hoe de wijzigingen tot nu toe zichtbaar zijn geworden (we laten het vak wiskunde D buiten beschouwing, aangezien dit vak geen centraal examen kent). Daarbij bekijken we wiskunde B op het vwo wat nauwkeuriger.

Havo wiskunde A

Op de havo zijn lineaire en exponentiële verbanden nog steeds belangrijke onderwerpen van wiskunde A. Kansrekening, voorheen toch een groot onderdeel van het programma, is volledig verdwenen. Hiervoor in de plaats is statistiek gekomen, evenals meer aandacht voor rekenregels en algebraïsche vaardigheden.

Leerlingen werden in het afgelopen examen bijvoorbeeld gevraagd om de formule $I_C = (216 - 2{,}84V + 1{,}12T) \cdot 0{,}97^V$ te herleiden naar de vorm $I_C = a \cdot T + b$ in geval dat $V = 0{,}43$. Een exercitie die wiskundig wellicht niet bijster interessant is, maar voor menig leerling bij havo wiskunde A toch een struikelblok kan vormen. Dergelijke opgaven kwamen voorheen ook al in beperkte mate in examens voor (meestal één per examen), maar nu lijken ze wat vaker te verschijnen.

Qua statistiek gaat het niet meer om het uitrekenen van een gemiddelde of mediaan, of om het tekenen van een histogram of boxplot, maar om het *redeneren* op basis van dergelijke informatie en het doen van statistische uitspraken. Zo kregen leerlingen in de opgave 'Lunchen' van het examen 2018 informatie over een onderzoek in de Verenigde Staten waarbij gekeken werd naar de hoeveelheden kilocalorieën die besteld werden door consumenten die in hun lunchpauze ergens gaan eten. Hiertoe werden in New York bij 167 willekeurig gekozen lunchrestaurants met calorie-informatie op hun website rond lunchtijd kassabonnetjes van klanten ingenomen. Leerlingen werd gevraagd om na te denken over waarom de uitspraak "Veel volwassenen nuttigen bij de lunch meer dan de aanbevolen hoeveelheid kcal" te algemeen is. Gedacht kan daarbij worden aan antwoorden zoals dat er alleen in New York gekeken is, dat er alleen is gekeken naar restaurants met calorie-informatie op hun website en dat er alleen is gekeken naar consumenten die hun lunch bij een restaurant halen.

Ook moesten leerlingen aan de slag met een relatieve cumulatieve frequentiepolygoon (waarbij op basis van bepaalde gegevens bepaald moest worden bij welk type restaurant deze hoorde) en werd hen gevraagd om "met behulp van het formuleblad" te bepalen of het lezen

van calorie-informatie effect heeft op de hoeveelheid calorieën in de bestelling. Dit genoemde formuleblad bevat enkele methoden om te bepalen of er verschil is tussen twee groepen: de *phi-coëfficiënt* (een speciaal geval van de correlatiecoëfficiënt van Pearson), het *"maximaal verschil in cumulatief percentage"* (waarbij gekeken wordt naar de maximale verticale afstand tussen twee cumulatieve frequentiepolygonen), de *effectgrootte* (het verschil tussen twee gemiddelden uitgedrukt in aantal standaardafwijkingen) en de *boxplot-vergelijking* (waarbij gekeken wordt naar de ligging van de boxen en de medianen). De examens, evenals eerdere pilotexamens, bevatten steevast opgaven waarin enkele gegevens verstrekt worden en leerlingen de juiste maat van het formuleblad moeten gebruiken. Het is de vraag in welke mate dit daadwerkelijk inzicht oplevert; voor velen zal het een kwestie zijn van pattern matching en invullen.

In het examen wiskunde A worden alle vragen binnen een context gesteld. Zo kregen de kandidaten in 2018 naast de genoemde restaurants te maken met brandgevaar, hemoglobinegehalte, aardbevingen, de BMR en de ecologische voetafdruk. Om de deelvragen eenduidig te stellen zonder de context onrecht aan te doen moeten de examenmakers nogal wat moeite doen, wat resulteert in lange stukken tekst waaruit kandidaten informatie opdiepen om vervolgens hun standaardberekeningen uit te voeren. Elke zin in de opdrachten staat er met reden, maar aan de kandidaten zullen de subtiliteiten vermoedelijk voorbijgaan.

Al met al is het de vraag of het doel van de herziening bij wiskunde A — leerlingen een kritischere houding aanmeten en beter voorbereiden op de statistiek van vervolgopleidingen — met dit programma en bijbehorende examens volledig behaald wordt. Het lijkt wat geforceerd om met onderwerpen uit de statistiek bezig te zijn, zoals de vuistregels van de normale verdeling en het bepalen van betrouwbaarheidsintervallen, zonder dat leerlingen ook maar enige ervaring hebben opgedaan met kansrekening. Het hangt natuurlijk ook af van de manier waarop scholen het schoolexamen inrichten (met daarin bijvoorbeeld het onderwerp 'Statistiek met ICT'). Het is spijtig dat er geen tijd is om een betere basis te leggen alvorens geavanceerdere onderwerpen uit de statistiek behandeld moeten worden.

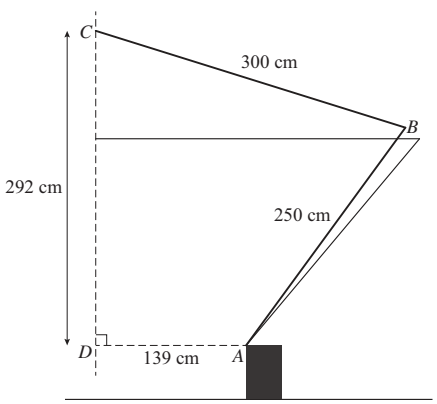
Havo wiskunde B

Bij havo wiskunde B zit de grootste wijziging in de meetkunde: ruimtemeetkunde heeft plaatsgemaakt voor analytische meetkunde. Geen aanzichten, uitslagen of doorsneden meer, en geen inhoud van afgeknotte piramides of cilinders. Leerlingen mogen nu aan de slag met afstanden en hoeken in het platte vlak en algebraïsche meetkundige methoden.

Een typische opgave uit de nieuwe domeinen in het eindexamen van 2018 betreft het deeldomein 'Algebraïsche methoden'. Het is een volledig contextvrije opgave waarin leerlingen gevraagd wordt te bewijzen dat de afstand tussen de lijn $l\colon y = \frac{3}{4}x + \frac{11}{4}$ en het punt $P(6,1)$ gelijk is aan 5. Voor leerlingen recht-toe-recht-aan, waarbij het niet duidelijk is waarom de uitkomst al gegeven wordt in deze opgave of waarom l niet gegeven wordt door $l\colon -3x + 4y = 11$.

Ook mochten leerlingen aan de slag met een cirkelvergelijking van een cirkel met middelpunt M (te vinden via kwadraatafsplitsen) om het verschil uit te rekenen van de afstand van M tot de x-as en de afstand van $P(6,1)$ tot M. Vreemd genoeg werd vooraf al vermeld dat de afstand van P tot de cirkel ook 5 is.

Ook nieuw was het toepassen van hulpmiddelen zoals de sinus- en cosinusregel en/of de stelling van Pythagoras (deeldomein 'Afstanden en hoeken in concrete situaties'). In het afgelopen examen werd dit onderwerp geëxamineerd in een opgave waarin de draagarmen van een hoogwerker vereenvoudigd zijn weergegeven (zie Figuur 1). De vraag was om het verschil te bepalen tussen hoek B in twee gegeven situaties, waarbij vooraf al gegeven was dat deze hoek $50°$ is in het geval dat de langste arm horizontaal is. De hoop is



Figuur 1 Plaatje uit het examen havo wiskunde B 2018.

dat leerlingen bij een rechte hoek al snel denken aan Pythagoras (en dus de benodigde hulplijn AC tekenen), en daarna goed getraind zijn om bij een driehoek met alle zijden gegeven de cosinusregel aan te roepen.

Vwo wiskunde A

Ook op het vwo heeft wiskunde A een aanzienlijke verandering ondergaan. Waar voorheen een groot deel van het examen bestond uit kansrekening (waaronder het gebruik van de binomiale en normale verdeling) en hypothesetoetsen, bevatten de nieuwe centrale examens niets meer van dat alles. Ook statistiek, nog wel onderdeel van de examens bij havo wiskunde A, maakt geen onderdeel meer uit van het centrale examen op het vwo. Overigens worden kansen en kansverdelingen, evenals verklarende statistiek, nog wel getoetst in het schoolexamen. Daarbij is er in vergelijking tot vroeger meer aandacht voor het gebruik van ICT en aspecten rondom probleemstelling en onderzoeksontwerp.

Het centrale examen bevat dus uitsluitend algebra en analyse, en een heel klein beetje combinatoriek. Nieuw in het examen zijn rijen en reeksen (nuttig ter voorbereiding op economische studies), evenals het gebruik van sinusoïden voor het beschrijven van praktische fenomenen (wellicht minder nuttig voor de gemiddelde A-leerling). Ook de e-macht en natuurlijke logaritme met hun afgeleiden zijn van de partij, maar kwamen dit jaar nog niet erg goed uit de verf in het examen.

Bij een zekere opgave wordt de zogeheten 'Shannon-index' gegeven als $H = -(p_1 \ln(p_1) + p_2 \ln(p_2))$. Leerlingen moeten voor een toepassing omtrent de diversiteit van een bosgebied, gegeven waarden voor p_1 en p_2, deze waarden invullen en de resulterende uitkomsten vergelijken. Geen enkele kennis van de natuurlijke logaritme is nodig om deze opgave te maken, afgezien van de positie van de bijbehorende knop op de grafische rekenmachine. Vervolgens wordt vereenvoudigd tot de formule $H = -(p\ln(p) + (1-p)\ln(1-p))$ en moeten leerlingen onderzoeken tot welke waarde H nadert als p naar 0 daalt. Een schets en bijbehorende conclusie voldoen; ook hier is dus geen specifieke kennis nodig. Als vervolgens moet worden bepaald voor welke waarde van p de Shannon-index maximaal is, wordt de afgeleide al gegeven — en dat terwijl het bepalen van de afge-

leide van e-machten en natuurlijke logaritmen nu juist sinds dit jaar bevraagd zou kunnen worden.

De goniometrische functies kwamen afgelopen jaar aan de orde in het centraal examen in de opgave 'Jaarringen', waarin de groeisnelheid van een boom gemodelleerd werd met een sinusoïde. De leerlingen werden uitgenodigd om aan de hand van gegeven eigenschappen over maximale en minimale groei en periodiciteit de groeifunctie G te schrijven in de vorm $G = a + b\sin(c(t-d))$.

Vwo wiskunde C

Waar bij wiskunde A differentiëren, e en de natuurlijke logaritme en goniometrische functies op het programma staan, kent wiskunde C twee eigen domeinen: 'Vorm en ruimte' en 'Logisch redeneren'. In de pilotexamens die afgelopen jaren werden afgenomen, leek het voor examenmakers nog wel eens zoeken naar deugdelijke opgaven over beide onderwerpen. In het eerste echte examen was het onderwerp van de opgave over vorm en ruimte een schilderij van Francis Bacon (zie Figuur 2). Het perspectief in het schilderij lijkt, uitgaande van een balkvormige ruimte, niet te kloppen. Dus was de opdracht om op een juiste manier een perspectieftekening van de situatie af te maken en de plek aan te geven van de figuur op een afstand van de halve diepte van de ruimte in diezelfde perspectieftekening.

Het domein 'Logisch redeneren' kwam aan de orde in de opgave over varianten van het cyrillische alfabet in Servië, Bulgarije en Rusland. Aan de hand van mededelingen zoals "Er zijn drie letters die alleen in Rusland voorkomen" en "Van de 39 verschillende letters komen er 24 voor in alle drie de landen" diende beredeneerd te worden hoeveel letters het Servische alfabet heeft. Vervolgens werden het Griekse, Russische en Latijnse alfabet vergeleken. Gegeven de alfabetten was de vraag onder meer voor welke letters geldt $G \wedge L \wedge \neg A$, waarbij A betekent dat de letter in alle drie de alfabetten voor komt.

Figuur 2 Plaatje uit het examen vwo wiskunde C 2018.

De twee genoemde opgaven toetsen de examenstof van beide domeinen, maar of ze illustratief zijn voor de oorspronkelijke bedoeling van het vernieuwde wiskunde C-programma is de vraag. De vorm en ruimteopdracht is een bescheiden uitbreiding van hetgeen er in de onderbouw bij tekenen al is geleerd en het logisch redeneren ontstijgt nauwelijks het oplossen van een raadsel, met een erg klein begin van formeel redeneren. Opvallend is verder dat nogal wat wiskunde C-opgaven een kunstwerk als context met zich meedragen. Dat daarmee aansluiting wordt gevonden met de belangstelling van de leerlingen is niet vanzelfsprekend. Aan het kiezen voor het profiel 'Cultuur en maatschappij' willen ook nog wel eens andere redenen ten grondslag liggen dan belangstelling voor kunst. De bedoeling van het vak is ook om gericht voor te bereiden op studies als rechten. In de praktijk kiezen leerlingen toch vaak voor wiskunde A omdat dat vak toegang geeft tot meer studies.

Vwo wiskunde B

Nieuw in het curriculum van vwo wiskunde B zijn limieten en continuïteit, en meer nadruk op de absolute-waardefunctie en de inverse functie. Net als bij de havo zit het grootste verschil hem in de meetkunde. Waar tot voorheen nog altijd uitgebreid aan bewijzen in de vlakke meetkunde werd gedaan, ligt de focus van het nieuwe programma op de *analytische meetkunde*. Leerlingen gaan aan de slag met vergelijkingen voor cirkels en lijnen, bepalen afstanden tussen punten, lijnen en cirkels, onderzoeken loodrechte stand, leren over vectoren, het inproduct en zwaartepunten, en gebruiken de sinus- en de cosinusregel. Bovendien behoren parameterkrommen inclusief baansnelheid en baanversnelling tot het programma. Dit alles overigens tweedimensionaal. Ook komen typische methoden uit de analytische meetkunde aan bod, zoals het kiezen van een assenstelsel en het handig kiezen van variabelen.

Onder verscheidene docenten heerste vooraf het idee dat de analytische meetkunde wat 'leerbaarder' is dan de synthetische meetkunde van voorheen. Het geven van bewijzen met koordenvierhoeken, omtrekshoeken en congruentiegevallen vereiste nogal eens enige creativiteit, en een leerling met weinig gevoel voor dergelijke zaken werd tot wanhoop gebracht door een stapel aan meetkundige opgaven. De nieuwe analytische meetkunde is in opzet wellicht wat meer gestructureerd — een leerling zou sneller kunnen ontdekken hoe aan een opgave te beginnen. Voor enkele typen opgaven is dat inderdaad het geval, maar de onderwerpen uit de analytische meetkunde blijken rijk genoeg te zijn om leerlingen echt aan het denken te zetten. Dat is prettig, omdat leerlingen daardoor enige creativiteit aan de dag moeten leggen en wat complexere denkstappen leren maken. Maar werd daarop ook een beroep gedaan tijdens het centraal examen van afgelopen jaar?

Uiteraard hebben de leerlingen hard geoefend met de pilotexamens, om zo een beeld te krijgen van het niveau. Deze pilotexamens waren veelal aan de lastige kant, in de zin dat ze een hoge N-term hadden (waardoor de cijfers werden opgehoogd ten opzichte van de reguliere normering). Zo bleek het nodig om in 2017 de N-term op 1,9 vast te stellen, maar liefst 0,9 punt op het cijfer cadeau, en kende 2016 zelfs een N-term van 2,4. Een echt cadeau is het in zo'n situatie voor de leerling overigens niet, aangezien een te lastig examen tijdens het maken ervan voor behoorlijke stress kan leiden — op dat moment weet de leerling immers nog niet dat er achteraf gecorrigeerd zal gaan worden.

Het lijkt dat de examenmakers met het eerste 'echte' examen vwo wiskunde B nieuwe stijl wat aan de veilige kant zijn gaan zitten — de N-term kwam uit op 0,9 en daar zijn de leerlingen op zich nog goed mee weg gekomen. Maar weinig opgaven deden een beroep op de 'wiskundige denkactiviteiten'. Veel sommen konden gemaakt worden op basis van routine (mits voldoende aanwezig) en met name de analytische meetkunde leidde niet tot grote problemen. Ook opvallend: het inproduct komt in de uitwerkingen in het geheel niet aan de orde.

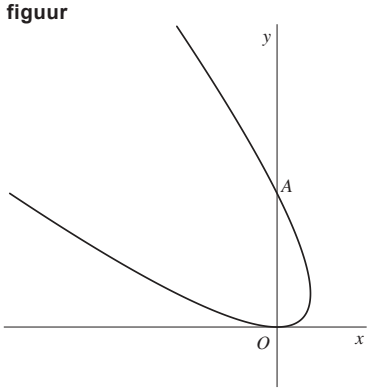
Het examen begon met een prettige binnenkomer, zoals weergegeven in Figuur 3. Leerlingen zijn bekend met de formule

De beweging van een punt P wordt gegeven door de volgende bewegingsvergelijkingen:

$$\begin{cases} x(t) = 1 - t^2 \\ y(t) = (1+t)^2 \end{cases}$$

In de figuur is de baan van P weergegeven.

figuur



De baan van P snijdt de y-as in de oorsprong O en in punt A. Zie de figuur.

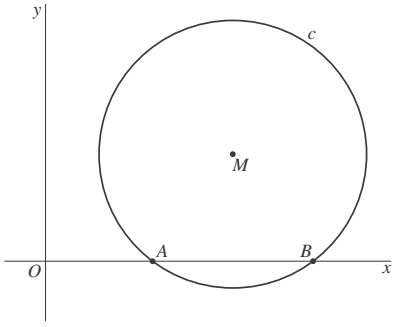
1 Bereken exact de snelheid waarmee P door punt A gaat.

Figuur 3 Opgave 1 van het examen vwo wiskunde B 2018.

Zwaartepunt en rakende cirkels

Gegeven is cirkel c met middelpunt $M(14, 8)$ en straal 10. Deze cirkel snijdt de x-as in de punten A en B met $x_A < x_B$. Zie figuur 1.
In A bevindt zich een puntmassa met massa 3, in B een puntmassa met massa 1 en in M een puntmassa met massa 2.

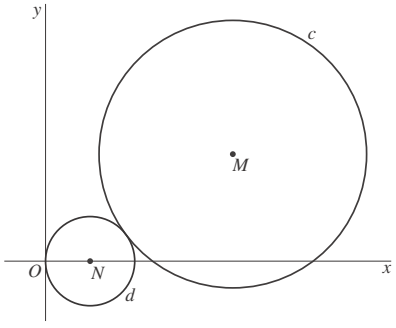
figuur 1



5p **6** Bereken exact de coördinaten van het zwaartepunt van deze drie puntmassa's.

De cirkel d met middelpunt N raakt de y-as in de oorsprong O en raakt cirkel c zoals weergegeven in figuur 2. Deze figuur staat ook op de uitwerkbijlage.

figuur 2



5p **7** Bereken exact de straal van cirkel d. Je kunt hierbij gebruikmaken van de figuur op de uitwerkbijlage.

Figuur 4 De opgave 'Zwaartepunt en rakende cirkels' uit het examen vwo wiskunde B 2018.

$v(t) = \sqrt{x'(t)^2 + y'(t)^2}$ en gelukkig behaalde de gemiddelde leerling landelijk 90% van de punten voor deze opgave.

De vervolgvraag, waarbij leerlingen moesten aantonen dat voor de coördinaten van ieder punt op de kromme geldt dat $(x+y)^2 = 4y$, bleek voor de meesten iets lastiger. Hoewel het in principe een eenvoudige invuloefening had kunnen zijn, gingen leerlingen in groten getale eerst t uitdrukken in x of y om vervolgens dat weer te substitueren in de andere vergelijking. Vaak werd daarbij echter $y = (1+t)^2$ herleid naar $\sqrt{y} = 1 + t$, waarbij dan $-\sqrt{y} = 1 + t$ over het hoofd gezien werd.

Interessant vanuit de curriculumvernieuwing is de opgave 'Zwaartepunt en rakende cirkels' (zie Figuur 4), bestaande uit twee vragen behorende tot de nieuwe examenstof. Bij vraag 6 konden leerlingen een vergelijking voor de cirkel opstellen, $y = 0$ invullen en dan berekenen waar A en B zich bevinden. Vervolgens hoeft nog slechts een gewogen gemiddelde van de coördinaten van M, A en B bepaald te worden. Met landelijk 81% van de punten gescoord, ook een eenvoudige opgave te noemen, waar weinig creativiteit voor nodig was. Een enkeling wist zelfs onder de cirkelvergelijking uit te komen door het punt $P(14,0)$ te tekenen en Pythagoras toe te passen in driehoek APM om zo te komen tot $AP = BP = 6$, waarna de coördinaten van A en B snel gevonden waren — een mooie aanpak, en het feit dat meerdere wegen naar Rome leiden is karakteristiek voor de analytische meetkunde in het nieuwe programma.

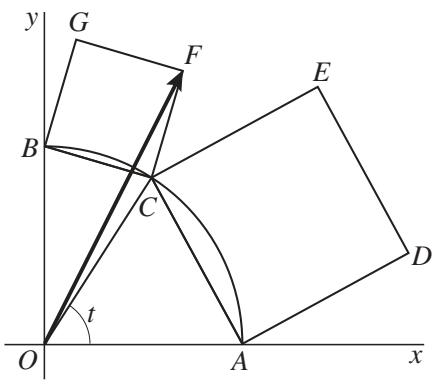
Vraag 7 ging door op dezelfde situatie en kon worden opgelost met een aanpak die leerlingen in aanloop naar het examen nogal eens voorbij zagen komen: kies een handig lijnstuk waarvan de lengte op twee manieren uitgedrukt kan worden in de gevraagde variabele en stel deze twee expressies aan elkaar gelijk. In dit geval noemen we de straal van cirkel d voor het gemak even r, en is vervolgens snel te zien dat $NM = r + 10$ en dat volgens Pythagoras bovendien geldt dat $NM = \sqrt{(14-r)^2 + 8^2}$. Gelijkstellen, kwadrateren en oplossen leidt tot $r = 3\frac{1}{3}$; landelijk werd op deze opgave gemiddeld 66% van de punten gescoord. De oplossing van deze opgave lag niet voor iedereen dicht onder de oppervlakte.

Het examen bevatte één opgave met een context, genaamd 'Sheffield Winter Garden'. Ter inleiding op deze context werd eerst

een vraag gesteld omtrent een kettinglijn, waarbij de lengte van deze lijn bepaald moest worden. Lastig leeswerk, maar uiteindelijk voornamelijk een kwestie van goed lezen en invullen. De opgave vervolgde met een vraag waarbij de boog getransformeerd wordt en waarbij leerlingen de juiste translaties en vermenigvuldiging moeten toepassen. Een verwarrende opgave, met wiskundig op zich geen spectaculaire aspecten.

Limieten en continuïteit kwamen aan de orde in een opgave waarin een functie met het beetje flauwe voorschrift $h(x) = \frac{\ln(\sqrt{x})}{\ln(x)}$ werd opgevoerd. Leerlingen moesten hierbij de coördinaten van de perforatie bepalen.

De laatste opgave was een meetkundeopgave over het plaatje in Figuur 5. Daarbij werd gegeven dat $0 < t < \frac{1}{2}\pi$ en dat C op de kwartcirkel door $A(1,0)$ en $B(0,1)$ ligt. Gevraagd werd naar de waarde van t waarvoor de oppervlakte van vierkant $ADEC$ twee keer zo groot is als de oppervlakte van vierkant $BCFG$. Voor de veiligheid werd voorafgaand aan de opdracht vermeld dat C coördinaten $(\cos(t), \sin(t))$ heeft. Daarmee werd de angel uit de opgave gehaald.



Figuur 5 Plaatje bij de laatste opgave van het examen vwo wiskunde B 2018.

Iets dergelijks geldt voor het allerlaatste onderdeel. Daarin werd gevraagd aan te tonen dat geldt $\overrightarrow{OF} = \begin{pmatrix} 1-\sin(t)+\cos(t) \\ \sin(t)+\cos(t) \end{pmatrix}$. Door het geven van een eindantwoord werd de opdracht voor goed ingevoerde leerlingen teruggebracht tot een routineklus zonder gevaar voor verwarring tussen rechts- of linksom draaien.

Het centraal examen wiskunde B op het vwo deed in het eerste tijdvak dus geen groot beroep op 'wiskundige denkactiviteiten'. De bedoeling van het nieuwe programma bij alle wiskundevakken van havo en vwo was ook om die activiteiten (te maken hebbend met probleemoplossen, abstraheren, manipuleren, redeneren en structureren) een grotere rol te laten spelen. De eerlijkheid gebiedt te zeggen dat het examen van het tweede tijdvak minder weggaf als het gaat om het vinden van een succesvolle oplossingsstrategie. Zoals wiskundige K.P. Hart opmerkte nadat hij uitwerkingen had gemaakt van beide examens [2]: qua wiskunde ontlopen de twee elkaar niet veel. In principe moeten vergelijkbare handelingen worden uitgevoerd, maar bij het herexamen kostte het iets meer moeite om een fijne eerste stap van een oplossing te vinden. Het resulteerde overigens in een N-term van 1,8 voor het tweede tijdvak.

Concluderend

Goedbeschouwd zijn de vakken wiskunde A en C in het centraal examen met het onvermoeibaar schakelen van de ene context naar de andere (wiskunde A havo zagen we al, voor wiskunde C is het rijtje contexten: windenergie, Francis Bacon, vermenigvuldigen op de handen, grauwe ganzen, het cyrillische alfabet, toren van achtvlakken) 'wiskundig denkactiever'. Helaas, zoals gezegd, zijn de contexten echter zodanig dichtgetimmerd om misverstanden te voorkomen dat een leerling uiteindelijk als een cavia in de *Wiekentkwis* via wegwijzers naar het hokje met een juiste oplossing wordt geleid.

Wel kennen de nieuwe wiskunde A-examens bij zowel havo als vwo een zogeheten onderzoeksopgave: een opgave waarmee meer punten te verdienen zijn omdat het oplossen ervan wat meer werk kost en ook niet via gebaande wegen plaatsvindt. Zo moest er bij de havo op basis van een behoorlijke hoeveelheid gegevens worden bepaald in welke jaren de ecologische voetafdruk onder een bepaalde hoeveelheid zal zijn, waarbij zowel lineair als exponentieel gerekend diende te worden. Bij het vwo gingen leerlingen aan de slag met grafieken over vermogen en snelheid in de context van een massasprint bij het wielrennen.

Wat wiskunde B betreft op het vwo is het nog even zoeken naar het gewenste niveau van complexiteit; getuige het verschil tussen eerste en tweede tijdvak, een nogal precair proces.

Referenties

1 J. Zwarteveen en N.C. Verhoef, Veranderingen in het eindexamenprogramma?, *Nieuw Archief voor Wiskunde* 5/14(4) (2013), 253–256.

2 https://hartkp.weblog.tudelft.nl.