Den nye maskinlæringen: Kunstig intelligens eller bare gode verktøy?

Jan Arne Telle, institutt for informatikk, universitetet i Bergen

Den nye maskinlæringen, særlig dype nevrale nettverk (DNN), skaper overskrifter om en kunstig superintelligens som truer menneskets eksistens. Vil menneskehetens skjebne i fremtiden ligge i algoritmene til slike maskiner? Skal vi forholde oss til denne type spørsmål trenger vi å forstå disse maskinene mennesket har skapt i sitt bilde.

«Hey Siri, send my sister a message saying I will be ten minutes late», sa sønnen min til sin iPhone ved middagsbordet. En kvinnestemme svarte «Who is your sister», så han la til «It is Aurora». Til min forbløffelse fikk han gjort det han ville uten å legge fra seg kniv og gaffel, og uten å involvere noen andre. I alle dager, hva blir det neste, tenkte jeg.

Troen på kunstig intelligens, på at maskiner kan programmeres slik at de framviser intelligens, er i våre dager formidabel. Statusen til dette fagfeltet har gått i bølger siden 1960-tallet, fra medieyndling til hakkekylling og tilbake, og i dag tipper optimismen over i bekymring. I sin bestselger om superintelligente maskiner, *Superintelligence: Paths, dangers, strategies* (2014), hevder filosofen Nick Bostrom at på samme måte som gorillaens skjebne som art i dag ligger i hendene på oss mennesker, vil menneskehetens skjebne i fremtiden ligge hos de superintelligente maskinene. Også fysikeren Stephen Hawking tror utviklingen av en fullstendig kunstig intelligens kan bety slutten på menneskeheten. Hva har endret seg siden forrige bølgedal? Har vi fått en ny grunnleggende forståelse av intelligens? Har vi utviklet nye algoritmer, eller er det bare maskinene som er blitt raskere? For å svare på disse spørsmålene må vi se nærmere på læring, særlig maskinlæring.

«Pornografi», sier Facebook og sensurerer bort et ikonisk bilde fra Vietnamkrigen. «Vov-vov!» sier min lille nevø henrykt fra barnevognen og peker på en katt. Under opplæring er det ikke uvanlig at eleven gjør feil. Min nevø har lært å skille vov-vov fra mjau gjennom trening på eksempler av begge typer, og selv om det for meg var tydelig at han så en katt, så var dette vitterlig en katt i bånd. På samme måte har bilde-sensuren til Facebook lært å skille vanlige bilder fra pornografiske, og selv om vi ikke synes det sensurerte bildet er pornografisk, så har det åpenbart en naken jente i fokus. Slike feil er en del av læringsprosessen. «Nei, mjau!» sier jeg til nevøen min, og korrigerer det jeg antar er hans hypotese om at et lite dyr i bånd er en vov-vov – han kan selvsagt ikke fortelle meg hva hans hypotese var. Men her er han ikke alene, også jeg har problemer med å redegjøre for hvordan jeg gjenkjenner en hund. Jeg kan redegjøre presist for hvorfor to ganger to er fire, og jeg vet nøyaktig hvordan jeg sjekker at et gangestykke er korrekt, men ikke hvordan jeg gjenkjenner en hund – det er noe jeg gjør til dels

ubevisst. Likevel, om noen viser meg et dyr og spør om det er en hund, vil jeg svært sjelden gjøre feil, til det har jeg sett for mange hunder og ikke-hunder.

Det er lett å programmere en datamaskin til å sjekke om et gangestykke er utført korrekt. Mennesker oppfant datamaskinen for å automatisere oppgaver som har en helt presis definisjon, som multiplikasjon av tall. Når vi ikke klarer å redegjøre presist for hvordan vi avgjør om et dyr er en hund eller ikke, virker det derimot merkelig at vi skulle klare å automatisere denne prosessen. Likevel har vi de siste årene, ved hjelp av den nye maskinlæringen, klart å automatisere prosesser som vi selv utfører ubevisst. En vesentlig del av denne automatiseringen skjer ved at maskinen bruker et enormt antall treningseksempler, med tilhørende fasitsvar, for å spisse hypotesen for begrepet som skal læres, for eksempel «hund». Etter endt trening anvender maskinen denne hunde-hypotesen på nye eksempler som dukker opp. Dersom eksempelet faller innenfor hunde-hypotesen, svarer maskinen at «ja, dette er en hund», hvis ikke svarer maskinen «nei, dette er ikke en hund». Svarene som gis viser seg å være svært nøyaktige, uten at konkrete forslag til hypotese for hva som er en hund ble gitt av programmereren.

Den beste maskinlæringen for mange applikasjoner kalles dyp læring og baserer seg på nevrale nettverk, ofte i form av det vi kan kalle «dype nevrale nettverk» eller DNN (LeCun, Bengio, Hinton 2016). De siste årene har DNN-læringen brutt stadig nye barrierer og blitt så avansert at beundring blandes med bekymring – tenk for eksempel på min reaksjon på Apples talende program Siri. For sortering av bilder i kategorier, som ulike hunderaser, har det kinesiske selskapet Baidu utviklet en læringsalgoritme basert på DNN som gjør færre feil enn et godt opplært menneske. Det mest slående med DNN-læringen er at den, bortsett fra fasitsvarene som gis under trening, ikke baserer seg på lærdom fra menneskelig ekspertise, ikke engang når det gjelder hvilke egenskaper (*features*) hypotesene skal bygge på. Det er nesten uforståelig at vi for eksempel kan trene opp evnen til å skille de forskjellige hunderasene uten å peke på hvilke egenskaper – som pels, snute, ører – maskinen skal basere en hypotese på. Det nye er altså ikke bare at vi har klart å automatisere prosesser vi utfører ubevisst, vi har også *automatisert selve læringsprosessen*. Man kan si at DNN-læring innebærer at maskinen lærer å lære, siden den selv oppdager de egenskapene en god hypotese baserer seg på.

I det populære kinesiske brettspillet Go er antall mulige kombinasjoner svært høyt, og de beste spillerne bruker i stor grad intuisjon. Derfor trodde man inntil nylig at en maskin ikke kunne programmeres til å spille Go på høyt nivå, i motsetning til sjakk, der Deep Blue slo Garry Kasparov allerede i 1997. Det var derfor en overraskelse for mange, også forskere innen feltet, da den DNN-opplærte maskinen AlphaGo fra selskapet DeepMind i fjor slo en av de beste Go-spillerne i verden (Silver mfl. 2016). Sjakkmaskiner går systematisk gjennom mulige trekk for å finne det beste, og vi

kan derfor i ettertid trekke ut en nyttig forklaring på valgene de tar. Hypotesen AlphaGo bruker for å gjøre et godt trekk er basert på statistisk materiale fra det enorme antallet treningskamper den har spilt, blant annet mot seg selv, og vi får ikke noen nyttig forklaring på valgene den gjør. Det kan altså se ut til at datamaskiner i dag ikke bare overgår oss i å følge regler, men også i å skaffe seg det vi kaller intuisjon, noe som kan forklare bekymringene til eksperter som Bostrom og Hawking.

I resten av dette essayet skal jeg forsøke å avmystifisere DNN-læringen og belyse mulighetene for kunstig intelligens (AI). Vi er langt fra å utvikle en kunstig *generell* intelligens (AGI) med fleksibilitet som et menneske, så jeg vil ikke fokusere på dette. Jeg skal heller ikke se på anvendelsene av DNN-læringen – selv om det er disse som er mest iøynefallende – men i stedet forklare, via relativt enkle modeller, hvordan denne teknologien fungerer. Mange vil ha et ord med i laget når disse teknologiene diskuteres, og da er det viktig å ha god bakgrunnsforståelse. Jeg skal presentere noen av de ideene og innsiktene som ligger til grunn for fagfeltet kunstig intelligens og for DNN-læring. En datamaskin er hierarkisk oppbygget og samspillet mellom nivåene er veldefinert. For en fullstendig forståelse må man sette seg inn i alle nivåene men det er det ikke plass til her. Et fokus på de lavere nivåene vil bidra til å avmystifisere. Siden en historisk gjennomgang vil åpne for en bredere forståelse, vil jeg gjøre dette kronologisk. Dermed må vi starte med grunnlaget for den moderne datamaskinen, nemlig ideene til den engelske informatikeren Alan Turing, en pioner innen kunstig intelligens.

Datamaskinens potensial i en enkel modell

Ideen om en maskin som kan utføre beregninger har dype historiske røtter, og arbeidet som har ført til dagens datamaskiner kan deles inn i to kategorier. På den ene siden har vi de grunnleggende ideene, som oftest utviklet av informatikere, matematikere eller logikere, og på den andre siden konkretiseringen av disse ideene i faktiske maskiner, utført av visjonære ingeniører i et industrielt og markedsstyrt kappløp. En moderne datamaskin er enormt komplisert, men den baserer seg på noen få, grunnleggende ideer.

Noen av de viktigste av disse ideene ble utviklet av Alan Turing for over 80 år siden (Turing 1936). På den tiden diskuterte man hvilke matematiske problemstillinger som kunne beregnes mekanisk, eller algoritmisk, altså gjennom en sammenhengende rekke veldefinerte steg. Mange algoritmer var kjent, men Turing var den første som ga en presis definisjon av hva en algoritme er. Han beskrev en maskin, eller hva vi kan kalle en modell for en datamaskin, og postulerte at denne «Turing-maskinen» kunne utføre enhver rekke av slike veldefinerte steg som ville falle innenfor det inntil da intuitive begrepet algoritme. Dette postulatet er i dag allment akseptert. En Turing-maskin kan utføre alle de beregninger

som kan gjøres av en datamaskin, både i dag og i framtiden. Man skulle dermed tro at den var meget komplisert, slik en datamaskin er, men det er den ikke.

Turing-maskinen er utviklet i analogi med en enkel modell for menneskets bevisste oppførsel og tar utgangspunkt i hva vi gjør når vi utfører et regnestykke med penn og papir: «Vi kan sammenligne et menneske som utfører en beregning med en maskin som bare har et endelig antall tilstander.» (Turing 1936: 231). Leseren anbefales å tenke gjennom dette selv, slik Turing gjorde for 80 år siden, og forsøke å forenkle og bryte ned i enkle bestanddeler det som skjer i vår bevissthet når vi for eksempel multipliserer to store tall:

Turing observerer at papiret ikke trenger å være to-dimensjonalt, vi kan tenke på det som en papirstrimmel med en lang rekke ruter. I hver rute kan man skrive et symbol. Idet vi starter multiplikasjonen står det for eksempel 123*68 i de seks første rutene. Vi starter med å gange 3 med 8. For å få til dette må vi først finne symbolet 3, det vil si høyre siffer i det første tallet. Turing innser at vi kun trenger være oppmerksom på ett symbol om gangen, så hans maskin har et 'lesehode' som ser én enkelt rute. Han bemerker også at vi alltid er i en bestemt tilstand. Vi starter med lesehodet i første rute, leser symbolet 1, er i tilstanden 'lete etter høyre siffer i det første tallet', og så lenge vi ikke ser symbolet * beveger vi lesehodet ett hakk til høyre og forblir i samme tilstand. I tillegg må vi kunne skrive nye symboler for å lagre halvferdige resultat, så lesehodet er et lese-skrive-hode. Turing bruker disse enkle mekaniske bestanddelene til å definere en maskin som opererer ved å bevege et lese-skrive-hode fram og tilbake, leser og skriver symboler, samtidig som den endrer sin tilstand. Selve 'hjernen' i maskinen, eller prosessoren, er en funksjon som gitt et symbol (det som leses av lese-skrive-hodet) og en tilstand (den maskinen er i for øyeblikket) vil utføre tre handlinger: skrive et bestemt symbol i ruten som lese-skrive-hodet står i, bevege dette ett hakk til høyre eller eventuelt til venstre, og endre til en bestemt tilstand. Så gjentas dette. Det er det hele! Det er ganske oppsiktsvekkende at noe som bygger på en modell med så ringe egenskaper kan få fremtredende filosofer til å snakke om menneskehetens mulige endelikt. Vi kan forenkle Turing-maskinen ytterligere ved å la den lese og skrive kun to symboler, 0 og 1. Så kan man enkelt kode andre symboler, for eksempel kan A, B, C og D kodes slik: A=00, B=01, C=10, D=11, slik at 001100 tilsvarer ADA.

Noe av det mest fundamentale og slående med datamaskinen er dens allsidighet. Dette er noe vi har vendt oss til og tar for gitt, men tenk for en sensasjon det hadde vært om vi på kjøkkenet plutselig bare trengte ett redskap i stedet for kjøleskap, ovn, brødrister, mix-master, kniv, brødfjøl og så videre. Datamaskinens allsidighet er en annen av Turings viktige observasjoner, og hans argument er igjen slående enkelt. Slik vi har beskrevet det hittil, vil vi trenge én Turing-maskin for å multiplisere tall, kall

den M, og en annen for å summere tall. Vi definerer M ved å skrive ned symbolene den opererer med, navnene på tilstandene den har, samt instruksjonene for 'hjernen'; vi må altså for hvert par bestående av et symbol og en tilstand angi hvilken handling den utfører. Gitt definisjonen av M og en input, for eksempel 123*68, kan vi følge instruksjonene for M og ende opp med resultatet 8364. Turing definerte en universell Turing-maskin, kall den U, som tar som input både 123*68 og definisjonen av M, og utfører den slaviske prosessen det er å regne ut resultatet 8364. Maskinen U kan simulere enhver annen maskin, og er derfor universell. Gir du U en input bestående av 23+6 samt definisjonen av en maskin S for å summere tall, så beregner U resultatet 29. Vi kan sammenlikne U med en datamaskin som kan programmeres ved bruk av M, S eller X til å bli en multiplikator eller en summator, ja endog til å 'bli egenarten' til en hvilken som helst algoritme X, for eksempel en algoritme for intelligens. Slik har den universelle Turing-maskinen U allsidigheten til en datamaskin, selv om brukervennligheten er elendig. Programmering ved hjelp av et høynivåspråk som Java eller Python er noe helt annet, med variabler og innebygde operasjoner, forgreninger (*if-then-else*), løkker og prosedyrekall.

Kunstig intelligens

Det finnes altså en algoritme for hver eneste oppgave en datamaskin potensielt kan utføre, og mange av disse algoritmene har vi ennå ikke oppdaget. Finnes det en algoritme for intelligens? I 1950 skriver Turing en artikkel hvor den første setningen lyder «I propose to consider the question, Can machines think?» (Turing 1950). Vi vet hva han mente med 'maskin' på dette tidspunktet, men han påpeker selv at ordet 'tenke' er vanskelig å definere, og ender det første avsnittet med å si at han derfor vil «erstatte spørsmålet med et annet, som er nært forbundet og uttrykt i relativt entydige ord». Han beskriver deretter det vi i dag kaller en «Turing-test», som kan formuleres på følgende vis: Kan en maskin vi kommuniserer med over nettet lure oss til å tro at den er et menneske? Hvis svaret er ja, mener Turing vi må vedgå at maskinen kan tenke. Turing-testen er imidlertid langt fra objektiv. For eksempel er en person lettere å lure jo mindre han vet om dagens teknologi. En god chatbot, altså et talende program som Siri, kan i dag lure mange som ikke følger utviklingen.

Noen, blant dem språkfilosofen John Searle, avviser Turing-testen og muligheten for en tenkende maskin fullstendig, men jeg er uenig. Searles argument om «Det kinesiske rom» er velkjent: Anta det finnes en algoritme som består Turing-testen på kinesisk. Anta tilsvarende at en person som ikke kan kinesisk sitter i et rom hvor han mottar spørsmål på kinesisk og deretter slavisk følger instruksjonene til denne algoritmen, som han er gitt i en regelbok skrevet på et språk han forstår, for tilslutt å sende svaret ut, igjen på kinesisk, og slik består Turing-testen. I følge Searle kan det ikke hevdes at denne personen faktisk forstår kinesisk, han bare manipulerer symboler slik regelboken instruerer ham, og følgelig kan

det heller ikke hevdes at en datamaskin som utfører algoritmen forstår kinesisk eller kan tenke. Maskinen kan bare simulere tenkning hvilket ikke er det samme som å tenke, hevder Searle. Som mange andre informatikere mener jeg dette argumentet er for antroposentrisk til å være avgjørende for spørsmålet om tenkning kan utføres av maskiner.

Det er ikke enighet om hva som er en god måte å måle intelligensnivået til en maskin (Hernandez-Orallo 2017). En IQ-test er for spesialisert, og relativt enkle program kan score høyt på deler av slike tester. På Turings tid ville nok mange lagt listen for lavt – man kunne for eksempel mene at den som slår verdensmesteren i sjakk måtte være tenkende og intelligent, selv om det var en maskin. Grensen for hva som skal til for å kunne definere noe som kunstig intelligens flytter seg dessuten stadig; straks teknologien bryter en ny barriere endrer man syn og sier at dette likevel ikke var den rette målestokken.

Perceptroner og AI-feltets bølgedaler

Siden bidraget til Turing, har forskningen på kunstig intelligens utviklet seg gradvis. Oppfatningen av fagfeltet har gått i bølgedaler, også blant myndigheter og investorer, grunnet forventninger som er skrudd i været for så ikke å innfris. I denne seksjonen vil jeg kort nevne noen relevante milepæler. Fagfeltet kunstig intelligens var i de tidligste årene opptatt av læring. I 1958 beskrev psykologen Frank Rosenblatt det han kalte et *perceptron*, og brukte dette til å utvikle en læringsalgoritme inspirert av nervevevet i hjernen (Rosenblatt 1958). Perceptronet er i seg selv meget enkelt, og gir et godt utgangspunkt for å forstå de minste bestanddelene i den moderne DNN-læringen, som vi skal se på i neste seksjon. Et nevron er en celle som mottar og sender signaler til og fra andre nevroner. Et perceptron modellerer en svært forenklet utgave av et nevron, ved å ta flere binære input og gi ett binært output, altså 0 eller 1, tenk på det som 'av' eller 'på'. Et perceptron har også *vekter*, tall som vektlegger hvor viktig hver input er. Bidraget fra en input er verdien (0 eller 1) ganget med vekten, og output er 1 dersom summen av bidragene overstiger en terskel. Slik kan et perceptron brukes til å ta en avgjørelse basert på flere fakta.

La meg gi et ganske trivielt eksempel. Anta du vil bestemme om du skal dra på skitur, og at tre faktorer spiller inn: er det godt vær, er føret bra, er du i god form? Dette representeres ved tre binære variabler x, y og z, slik at x=1 hvis været er godt, y=1 hvis føret er bra og z=1 hvis du er i god form, men ellers er de 0. Anta du er glad i en skitur, og hvis været er godt gjør det ingenting at det er kladdeføre og at du er bakfull. Vi bruker et perceptron og gir vekt 5 til været, vekt 2 til føret og vekt 3 til formen. Om vi setter terskelen til 5, får vi output 1 så lenge været er godt (siden 1*5+y*2+z*3 er minst 5). Vi tolker

output 1 som at vi skal gå på skitur. Når været ikke er godt (x=0) går vi kun på skitur dersom det både er bra føre og vi er i god form (siden 0*5+1*2+1*3=5). Om vi endrer vektene og terskelverdien vil perceptronet vektlegge faktorene annerledes og gi et annet grunnlag for avgjørelsen.

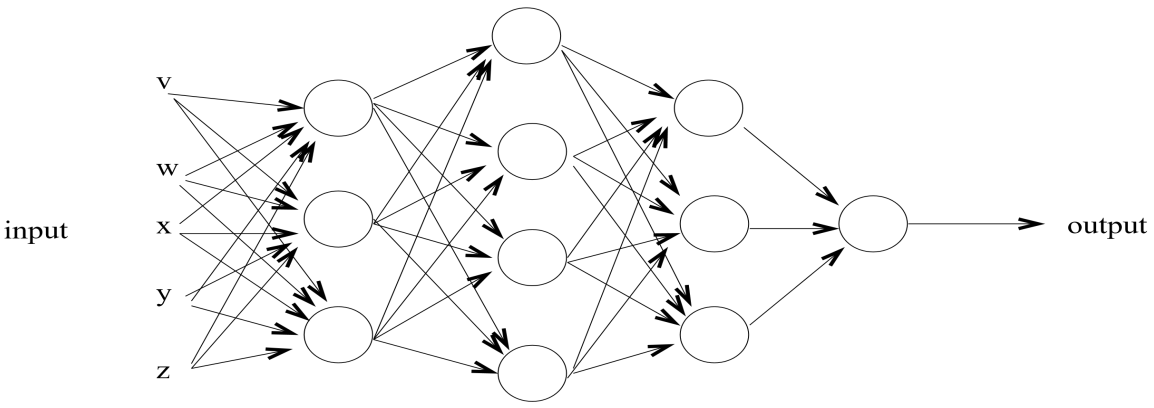
Rosenblatt var god til å markedsføre sin forskning – søk etter videoen «Perceptron research 50» for et interessant eksempel – men han lovet mye mer enn det var grunnlag for på den tiden. I 1969 ga Marvin Minsky og Seymour Papert ut boka *Perceptrons*, som tok opp perceptronets begrensninger (Minsky og Papert 1969). Dette bidro til at vi på 1970-tallet fikk den første såkalte AI-vinteren (kunstig intelligens-vinteren), preget av lite forskningsmidler til fagfeltet.

På begynnelsen av 1980-tallet snudde populariteten idet ekspertsystemene ble populære innen AI. Disse baserer seg ikke på læring, men på innhenting av faktakunnskap fra fageksperter. Faktaene og deres logiske sammenhenger blir samlet og kodet på en slik måte at nye fakta kan utledes automatisk. Ekspertsystemer ble med hell anvendt på flere områder, særlig innen business, og store investeringer ble gjort av ledende selskaper. Men ekspertsystemenes popularitet kollapset når heller ikke de levde opp til forventningene som var skapt av både forskerne selv og media, og en ny AI-vinter satt inn mot slutten av 1980-tallet.

Ideene til Rosenblatt ble holdt i live og videreutviklet gjennom hele denne perioden og utgjør i dag grunnlaget for de dype nevrale nettverkene. Anvendelsen av de nye maskinlæringsalgoritmene har de siste fem årene på nytt ført til medieoppslag med meget høye forventninger for fagfeltet kunstig intelligens.

Den nye maskinlæringen

Et kunstig nevralt nettverk er bygget opp av sammenkoblede kunstige nevroner, som vi for enkelhets skyld kaller nevroner. Hvert nevron kan være et enkelt perceptron. Nettverket i figuren har fire lag med nevroner, ordnet vertikalt med hvert nevron illustrert som en sirkel, med input til venstre og der det fjerde laget til høyre er output-laget, i dette tilfelle et enkelt nevron. Output-laget kan bestå av mange



nevroner – i et nevralt nettverk for å gjenkjenne håndskrevne siffer fra 0 til 9 vil det for eksempel være naturlig å ha ti nevroner i output-laget, ett for hvert siffer. De tre nevronene i det første laget gjør hver sin enkle avveining av de fem input-verdiene (v, w, x, y, z), og sprer sin output til hver av de fire nevronene i det andre laget. Hvert nevron i det andre laget gjør en avveining av de tre resultatene fra det første laget, og kan dermed ta en beslutning på et mer komplekst og abstrakt nivå enn nevronene i det første laget. På neste nivå kan enda mer komplekse beslutninger bli tatt.

For eksempel, for å klassifisere bilder fra piksler gitt som input kan første laget detektere noe så enkelt som kanter i bildet (overganger mellom to farger), neste lag samlinger av kanter som tilsammen utgjør en form, neste lag samlinger av former som tilsammen utgjør et øye, og siste lag noe så avansert som et fjes. Vektene settes automatisk gjennom trening og avgjør slik hvilken betydning de forskjellige lagene har i den hypotesen det ferdige trenede nettverket utgjør. Slik kan et nevralt nettverk sies å oppdage de egenskapene som en endelig hypotese baserer seg på, for eksempel for klassifisering av bilder. Når man snakker om dyp læring eller et dypt nevralt nettverk (DNN) mener man et nevralt nettverk med flere lag, som kan gjennomføre svært sofistikerte beslutningsprosesser.

Når man trener et slikt nettverk gir man et eksempel som input, og dersom output ikke er den ønskede, endrer læringsalgoritmen automatisk vektene og tersklene slik at output blir riktig. For å ikke forstyrre læringen fra tidligere treningseksempler bør en liten endring i en vekt føre til en liten endring i output. Derfor er det bedre å bruke det som kalles et *sigmoid-nevron* enn et perceptron. Sigmoid-nevroner har fortsatt vekter, men de tar som input og gir som output et reelt tall mellom 0.0 og 1.0, og output bestemmes av en glatt funksjon, som ikke gjør noen abrupte sprang slik perceptronet gjør fra 0 til 1 idet man treffer terskelen nedenfra, men hvor output-verdien istedet gradvis økes. I et nettverk for å gjenkjenne håndskrevne siffer kan input bestå av 900 piksler, tall mellom 0.0 og 1.0 som indikerer hvor mørk hvert av de 30*30 punktene er, som bildet med sifferet er delt inn i. Et treningseksempel vil være 900 piksler som utgjør et bilde av et siffer, for eksempel et 5-tall som er slurvete skrevet men likevel gjenkjennelig for et menneske, og hvis output-nevronet for 5 ikke har høyest verdi, så endres vektene. Etter endt trening vil nettverket ha utviklet en hypotese for gjenkjenning av siffer, som så kan anvendes på nyskrevne siffer.

Algoritmen som automatisk endrer vekter og terskelverdi krever mye matematikk. Den kan for eksempel basere seg på *Stochastic Gradient Descent* med *Backpropagation*. La oss få et inntrykk av hva som skjer. For hvert treningseksempel vil vi at nettverket gir rett svar, og hvis det ikke gjør det, tenker vi oss at vi betaler en bot, for eksempel en bot på en enhet for ett enkelt feil svar. Målet er at vekter og terskler settes slik at summen av alle bøter, kall denne verdien SB, minimeres. Vi starter i et

output-nevron og ser på hvordan en veldig liten endring i dets vekter påvirker SB. Hver vekt kan enten økes litt eller senkes litt, og endringen i hver vekt gjøres i retningen som senker SB, derav *Gradient Descent*. Så gjentas dette, inntil vi ikke kan få en lavere SB. Antallet treningseksempler er som regel veldig høyt, så i stedet for å beregne SB nøyaktig, vil vi i hver runde basere oss på kun en liten, vilkårlig mengde av dem, derav *Stochastic*. Dette er samme prinsipp som brukes for meningsmålinger, der et vilkårlig utvalg av noen få mennesker er nok til å gi god kvalitet. Slik kan *Stochastic Gradient Descent* brukes til å endre vektene i et output-nevron. *Backpropagation* er en metode som propagerer disse vekt-endringene bakover i nettverket, helt til vektene endres også i det første laget, slik at nevronene i hele nettverket samarbeider om å modellere treningseksemplene.

Varianter av dype nevrale nettverk

Variantene av DNN er mange, avhengig blant annet av hvordan man kobler lagene med nevroner sammen, se for eksempel nettstedet *The Neural Network Zoo* (Veen 2016) for en oversikt over 26 navngitte nettverkstyper. En av de viktigste variantene er *Recurrent Neural Networks* (RNN). Hittil har vi antatt at hvert enkelt treningseksempel er uavhengig av de andre, men i mange situasjoner gjelder ikke dette, for eksempel er rekkefølgen av ordene viktig i oversettelse av tekst. I et RNN er output fra ett lag av nevroner del av input til samme lag, hvilket skaper et slags minne som fører til at nettverket modellerer treningseksemplenes rekkefølge. Et RNN kan trenes på en tekstkorpus tegn for tegn og senere brukes til å angi sannsynligheten for neste bokstav, gitt en halvferdig setning. Forskeren Andrew Karpathy brukte denne metoden til å trene et RNN med tre lag og 512 nevroner i hvert lag på Shakespeares samlede verker i noen timer (Karpathy 2015). Deretter lot han nettverket generere ny Shakespeare-liknende tekst ved hele tiden å gjøre utvalg for neste tegn (bokstav, komma, linjeskift og annet) med den gitte sannsynligheten. Ikke bare genereres ord som er riktig skrevet, for menigmann kan det se ut som ekte vare. Om dette bare er et morsomt verktøy og ikke kunstig intelligens, så er det uansett imponerende. Her er et eksempel på slik Shakespeare-liknende tekst (Karpathy 2015):

```
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

En anvendelse av DNN som har fått mye blest i det siste, er dyp forsterkende læring (*Deep Reinforcement Learning*). Hittil har vi antatt at eksemplene kommer med fasitsvar, men i mange anvendelser av DNN, for eksempel for å lære dataspill eller Go, brukes i stedet forsterkende læring

(Mnih mfl. 2016). Her gis belønning for riktig oppførsel, men belønningen kommer forsinket og kun en sjelden gang. Læringen foregår ved at det spilles, og i starten velges trekkene vilkårlig i alle stillinger. Vinnes et spill, øker sjansen for at de trekkene som ga belønning blir valgt neste gang samme stilling oppstår. Underveis i læringen er antall mulige stillinger altfor mange til at man kan lagre dem i en vanlig tabell. I stedet brukes et DNN som en tilnærming til en slik tabell (*function approximator*). Input til DNN er stillingen, mens output, eller den ønskede output, vil være trekket som er foreslått basert på tidligere spill. Først henter man ut det foreslåtte trekket for en gitt stilling og gjør dette trekket. Basert på belønningen trener man så nettverket på de stillinger som påvirkes, ved å endre vekter i retning det ønskede output-trekket.

Forskeren og forfatteren Michael Nielsen har på sin hjemmeside michaelnielsen.org lagt ut en lærebok som har gitt inspirasjon til dette essayet. Nielsen skriver: «Det å forstå nevrale nettverk fullt ut er et problem som ... tester grensene for menneskets intellekt.» (Nielsen 2015, Kap. 3). Teorien for DNN-læring er kanskje ikke matematisk tilfredsstillende, men det vesentlige er at DNN-læringen fungerer i praksis. Det er viktig å påpeke at under DNN-læring vil de nevrale nettverkene og nevronene simuleres på vanlige datamaskiner, som i våre dager er ekstremt raske. Grafikkprosessorer (GPU), hvis utvikling i stor grad drives av det enorme markedet for dataspill, brukes til å parallellisere algoritmene, altså å utføre algoritmen på mange prosessorer samtidig, slik at de utføres enda raskere. Suksessen til DNN-læringen er avhengig av at nettverket kan trenes på veldig mange eksempler, noe som krever raske maskiner i tillegg til effektive algoritmer. Det kreves også tilgang til datalesbare treningseksempler. DNN-læring for oversettelse mellom to språk krever tekst presist oversatt av mennesker. Google Translate baserte seg opprinnelig på store mengder EU-dokumenter som var oversatt til 23 språk. I andre tilfeller er vi alle med på å skape treningsgrunnlaget for DNN-læringen, gjennom vår daglige bruk av nettbaserte applikasjoner der vi legger igjen datalesbare avtrykk. Dette burde ikke komme som en overraskelse; Google og Facebook er gratis fordi brukeren selv er produktet som både melkes og selges.

En maskin i vårt eget bilde

Ikke alle lar seg imponere av maskinlæringen. Lingvisten Noam Chomsky hevder slike statistiske metoder anvendt på store mengder data ikke gir den innsikt en vitenskap søker. Han sier at fagfeltet «... oppnår suksess på en veldig merkelig måte ... som jeg tror er ny i vitenskapshistorien. En tilnærming av uanalyserte data tolkes som suksess.» (Chomsky 2011). Peter Norvig, en nestor innen anvendelser av kunstig intelligens, har svart på innvendingene til Chomsky. Essensen i hans svar er at maskinlæring i praksis fungerer bedre enn noe annet, og ethvert vitenskapssyn må ta hensyn til dette

(Norvig 2012). Det er liten tvil om at DNN-læringen fungerer oppsiktsvekkende godt – Siri, Google Translate, AlphaGo er 'levende bevis' på dette – men hvor skal det ende?

Med en omskriving av første Mosebok kan vi si at mennesket har skapt en maskin i sitt bilde. Som vi har sett, baserer definisjonene til både Turing-maskinen og de nevrale nettverkene seg på menneskets selvforståelse. Denne vår skapning har allerede endret vårt syn på oss selv. Innen kognitiv psykologi er den dominerende modellen at mennesket prosesserer informasjon på en måte som likner datamaskinens. Noen mener sågar at utviklingen mot en kunstig intelligens, på toppen av vårt informasjonshungrige og nettverkstilknyttede samfunn, utgjør et mulig vannskille for menneskets posisjon i verden. Filosofen Luciano Floridi hevder at med Turing startet den fjerde revolusjonen i vårt syn på oss selv som unike skapninger (Floridi 2014). Ikke nok med at vi mennesker bor på en planet som ikke er i sentrum av universet (Copernicus og den første revolusjonen), eller at vi ikke er skarpt adskilt fra de andre dyrene på jordkloden (Darwin og den andre revolusjonen), eller at vi ikke er helt rasjonelle vesener med mulighet for fullstendig selvinnsikt (Freud og den tredje revolusjonen), Floridi mener vi etter Turing må innse at vi ikke er de eneste analytiske vesener med språklige evner men at vi deler denne egenskapen med andre informasjons-prosesserende organismer.

Det er meget mulig mennesket kan bli forbigått av maskiner på flere områder, og ikke bare i spill som sjakk og Go. Især DNN-læringens inntog på arenaen for det vi kaller intuisjon kan ha store konsekvenser. Den dystopiske teknologideterminismen, som kaller dette en trussel for menneskeheten og sier robotene vil overta, må ikke få skygge for de konkrete forholdsregler vi uansett burde ta. La meg avslutte med å nevne tre. For det første: jo flere som deler kunnskap om hvordan den kunstige intelligensen fungerer, jo lettere vil det være å føre en fruktbar debatt om dens mulige konsekvenser. For det andre: skal disse maskinene brukes fornuftig, bør vi sørge for at den åpne og frie akademiske forskningen fortsetter å styre utviklingen, og at den ikke overlates til et oligopol av store selskaper. For det tredje: på mange felt hvor maskiner faktisk tar over, skyldes dette hovedsakelig en økende byråkratisering som infiltrerer stadig flere av livets områder. Slik gjør vi oss selv overflødige. I en naiv tro på at vi kan skape feilfrie og objektive systemer, undergraver vi betydningen av vår subjektive dømmekraft.

Referanser

Bostrom, N. (2014). *Superintelligence: Paths, dangers, strategies*. Oxford University Press.

Chomsky, D. (2011). *Transcript from keynote panel Brains, Minds, Machines at MIT.* http://languagelog.ldc.upenn.edu/myl/PinkerChomskyMIT.html. [Sist lastet ned xx.xx.xx].

Floridi, L. (2014). *The fourth revolution*. Oxford University Press.

Hernandez-Orallo, J. (2017). *The measure of all minds: Evaluating natural and artificial intelligence*. Cambridge University Press.

Karpathy, A. (2015). *The unreasonable effectiveness of recurrent neural networks*. http://karpathy.github.io/2015/05/21/rnn-effectiveness/. [Sist lastet ned xx.xx.xx].

LeCun, Y., Y. Bengio og G. Hinton (2016). «Deep Learning». Nature 521, s. 436–444. DOI: https://dx.doi.org/10.1038/nature14539.

Minsky, M og S. Papert (1969). *Perceptrons.* MIT Press.

Mnih, V. mfl. (2016). «Human-level control through deep reinforcement learning». *Nature* 518, s. 529–533.

Nielsen, M. (2015). *Neural networks and deep learning*. Determination Press. http://neuralnetworksanddeeplearning.com/

Norvig, P. (2012). «On Chomsky and the Two Cultures of Statistical Learning». *Norvig.com*. http://norvig.com/chomsky.html. [Sist lastet ned xx.xx.xx]

Rosenblatt (1958). «The perceptron: A probabilistic model for information storage and organization in the brain». *Psychological Review* 65 (6), s. 386–408. DOI: https://dx.doi.org/10.1037/h0042519.

Silver, D. mfl. (2016). «Mastering the game of Go with deep neural networks and tree search». *Nature* 529, s. 484–489. DOI: https://dx.doi.org/10.1038/nature16961.

Turing, A. (1936). «On computable numbers, with an application to the Entscheidungsproblem». *Proceedings of the London Mathematical Society* 42 (1):230-265.

Turing, A. (1950). «Computing machinery and intelligence». *Mind* 59, s. 433–460. DOI: https://dx.doi.org/10.1093/mind/LIX.236.433.

Veen, F. (2016). «The neural network zoo». *Asimovinsitute.org*. http://www.asimovinstitute.org/neural-network-zoo/. [Sist lastet ned xx.xx.xx].