# Introduktion til Neurale Netværk

*Benny Lautrup*

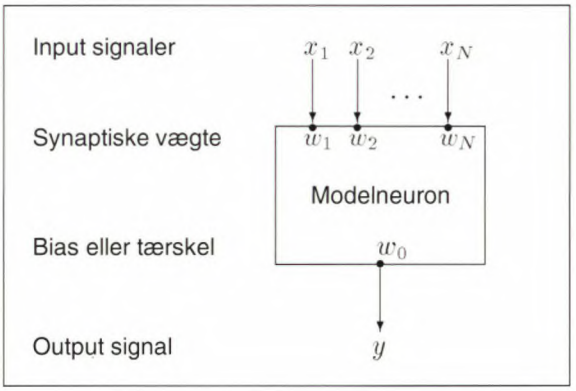
Det neurale netværk, vi alle besidder, er uden overdrivelse overordentligt kompliceret. Det består af omkring $10^{11}$ neuroner, som hver især har cirka $10^4$ synapser med andre neuroner, altså i alt omkring $10^{15}$ synapser. Det er efterhånden slået fast, at en vigtig del af hukommelsen hidrører fra, at synapsernes transmissionsegenskaber kan påvirkes af de signaler, der går igennem dem. Synapserne udviser en vis adaptiv plasticitet, en lokal erindring. Men det er ganske ukendt, hvorledes en global oplevelse omsættes til lokale synaptiske modifikationer, og omvendt hvorledes myriader af lokale synaptiske "potentialer" fører til en global erindring.

Distributionen af en global erindring på et utal af lokale synaptiske modifikationer kan meget vel umuliggøre en direkte eksperimentel tilgang til mekanismerne. Mange lokale og mange overordnede globale egenskaber kan naturligvis udforskes, men selve den proces, hvorved det mikroskopiske bliver makroskopisk er muligvis underlagt et "komplementaritetsprincip", der forhindrer os i at trænge dybt ind i den. Hvis vi forsøger at afdække hukommelsens mikroskopiske struktur, sker det på bekostning af de makroskopiske erindringer.

Det er blandt andet af denne årsag, at matematiske og fysiske modeller af neurale netværk kan være nyttige. Men når først modellerne er opstillet, får de deres eget liv og udvikler sig videre i deres egne baner. Ingeniørerne behøver ikke at holde sig til naturen, når de udvikler maskiner, men kan frit lade fantasien råde. Hvis de ikke havde den frihed, ville vi alle blive frygteligt søsyge af at flyve i luftfartøjer, der baskede med vingerne!

De matematisk-fysiske modeller for neurale netværk går tilbage til et arbejde fra 1943 af psykologen, neurovidenskabsmanden, filosoffen, og poeten Warren McCulloch og matematikeren Walter Pitts. De viste, at et netværk af meget simple matematiske neuroner — som dog havde en vis lighed med den tids fysiologiske modeller — kunne bringes til at udføre enhver logisk beregning, når bare netværket var tilstrækkeligt stort. Men beviset var matematisk og temmelig uigennemtrængeligt, og kunne ikke umiddelbart bruges til at implementere en vilkårlig logisk funktion.

I 1958 kombinerede Frank Rosenblatt disse ideer med en neural indlæringsmekanisme foreslået i 1949 af psykologen Donald Hebb. Det førte til Perceptronen, en "maskine" med to lag neuroner og et lag synapser. Denne maskine kunne bringes at udføre ønskede logiske funktioner gennem en iterativ proces, en "træningsproces" hvorved den "lærte" hvad den skulle gøre. Men ikke alle funktioner var mulige på grund af perceptronens specielle konstruktion. Kun en lille procentdel af det totale antal logiske funktioner kunne udføres. Det førte i 1969 til, at Marvin Minsky og Seymour Papert kritiserede hele området og i det væsentlige gennem deres indflydelse fik lukket af for pengene til det.




**Figur 1.** Modelneuronen danner den vægtede sum af de indkommende signaler $x_i$ og vægtene $w_i$. I denne skitse tilknyttes vægtene de enkelte neuroner og optræder derved som en slags lokal hukommelse (normalt tilknyttes de forbindelserne). Herved undgår neurale netværk den berømte von Neumann flaskehals mellem processering og hukommelse. Den vægtede sum modificeres af en ikke-linearitet, hvis placering afhænger af biaset eller tærsklen $w_0$.

Først i 1982-85 kom der igen liv i de kunstige neurale netværk gennem arbejder af John Hopfield, Rumelhart, McClelland, Hinton, Sejnowski og mange andre. Nye algoritmer til træning af vilkårlige netværk blev opfundet, og den kritik, der havde lammet feltet, faldt til jorden med et brag. Nye anvendelser så dagens lys gennem hurtige computersimuleringer og overbeviste mange om, at her var der muligheder for at udføre de mønstergenkendelsesopgaver, som sædvanlige programmeringsmetoder let kommer til kort overfor.

## Modelneuroner

Den matematiske modelneuron, som anvendes mest i dag, har en meget simpel struktur som vist i figur 1. Gennem sine forbindelser, "synapser", påvirkes modelneuronen af et vist antal, lad os sige $N$, indgangssignaler repræsenteret ved tallene $x_i$ $(i = 1, \ldots, N)$, som for eksempel kan komme fra andre neuroner eller fra omverdenen. Hver synapse beskrives ved en vægt $w_i$, som udtrykker synapsens transmissionseffektivitet overfor indgangssignalet. Det samlede signal $z$ som ankommer på modelneuronen beregnes som den vægtede sum over alle indgangssignalerne plus en konstant $w_0$, der udtrykker neuronens eget "bias"

$$z = \sum_{i=1}^{N} w_i x_i - w_0 \qquad (1)$$

Det er klart at biaset kan opfattes som den synaptiske vægt til en 0'te neuron, hvor signalet altid er $x_0 = -1$. Denne måde at behandle biaset på bevirker, at vi ikke i det følgende behøver at omtale det eksplicit, men blot kan forestille os, at alle aktive neuroner er koblet til den 0'te.

Endelig beregnes udgangssignalet som en i almindelighed ikke-lineær afbildning af dette signal

$$y = g(z) = \frac{1}{1+e^{-z}} \tag{2}$$

Et typisk valg er som vist her baseret på en eksponentialfunktion. Funktionen giver 0, når $z$ bliver meget negativ og vokser til 1, når $z$ bliver stor og positiv. og udtrykker en overgang mellem to forskellige neurontilstande, den inaktive og den aktive. Man siger ofte at $z$-værdierne bliver kvast af afbildningen. Læg mærke til, at fordi kvasefunktionen rejser sig skarpt i $z = 0$, kan $w_0$ opfattes som *tærskel* for den kunstige neurons aktivitet.

### Netværk

Modelneuroner kan kobles sammen i netværk. I figur 2 ses et netværk med to input, to *skjulte* neuroner og en enkelt output-neuron. Sådanne netværk uden feedback med tre lag neuroner (og to lag vægte) har vist sig meget nyttige i praktiske anvendelser, fordi de faktisk kan lære enhver ikke-lineær afbildning. Hvis man som i figur 2 betegner inputneuronernes aktiviteter med $x_k$, de skjulte neuroners med $s_j$, og outputneuronernes med $y_i$ $(i = 1, 2, \ldots, M)$ kan de fuldstændige sæt netværksligninger skrives

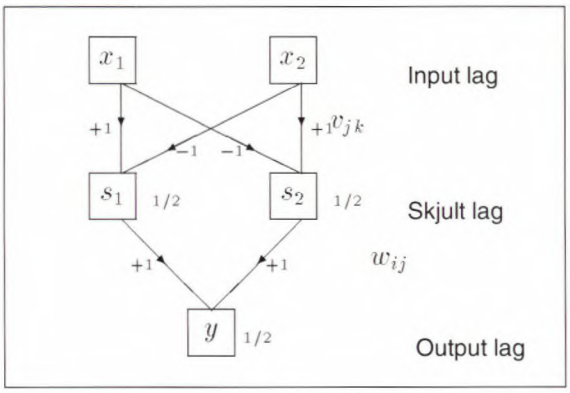
$$s_j = g\left(\sum_k v_{jk} x_k\right) \tag{3}$$

$$y_i = g\left(\sum_j w_{ij} s_j\right) \tag{4}$$

hvor det første vægtlag betegnes med $v_{jk}$ og det andet med $w_{ij}$. Tærskelværdierne er inkluderet i summerne. På denne måde bliver netværkets output en funktion af input og vægte, $y_i = F_i(x, v, w) = g(\sum_j w_{ij} g(\sum_k v_{jk} x_k))$. Neuronerne i inputlaget udfører faktisk ingen beregning, men bruges blot som hukommelsesregistre for inputaktiviteterne.

### Backpropagation

Man kan nu spørge om, hvorledes vægtene (og tærsklerne) skal vælges, for at sikre en ønsket opførsel af netværket. Hvis $(x, y)$ er et eksempel på input og tilsvarende output, er det jo ingenlunde givet, at netværkets faktiske output, $F_i(x, v, w)$ er lig med det ønskede. Der vil oftest være en afvigelse eller fejl $y_i - F_i(x, v, w)$. Den samlede kvadratiske afvigelse i output-laget beregnes derefter og for et givet eksempel er denne fejl en funktion af vægtene. Ideen er nu at finde et sæt vægte som gør fejlen mindst mulig. På grund af kvasefunktionens differentiable og glatte form vil fejlen være en differentiabel og glat funktion af vægtene, og det er derfor muligt at bruge *gradientnedstigning* til at finde et minimum. Denne metode tager sit udgangspunkt i enhver skiløbers erfaring, at hvis man vil nedad skal man gå i den modsatte retning af gradienten, der jo viser, hvor det går stejlest opad.




**Figur 2.** Dette netværk kan med de angivne vægte og tærskler udføre den vigtige logiske funktion $y = x_1$ **XOR** $x_2$. Der findes uendelig mange forskellige netværk som kan udføre netop denne logiske funktion, men de skal alle have *skjulte* neuroner, som hverken bærer input eller output. Det kan vises, at et netværk med $2^n$ neuroner i et skjult lag kan udføre enhver logisk funktion med $n$ input. Dette netværk er et typisk *feedforward* netværk, hvor informationen kun strømmer en vej fra input mod output. Det er også muligt at lave netværk med *feedback*, hvor information for eksempel kan føres fra output tilbage mod input.

En smule analytisk beregning viser, at denne metode fører til følgende udtryk for vægtændringerne i netværket

$$\Delta w_{ij} = \eta \delta_i s_j \tag{5}$$

$$\Delta v_{jk} = \eta \epsilon_j x_k \tag{6}$$

hvor $\delta_i = y_i'(y_i - F_i(x, v, w))$ er proportional med fejlen i output-laget, og $\eta$ er en positiv parameter, der kontrollerer, hvor hurtigt man stiger ned. Typiske værdier af denne parameter er 0.01–0.1, afhængigt af problemet. Størrelsen $y_i' = g'(\sum_j w_{ij} s_j)$ beregnes på samme måde som $y_i$, blot med den afledede af kvasefunktionen. De to udtryk har ganske ens struktur og $\epsilon_j$ kan derfor betragtes som den effektive fejl på det skjulte lag. Den beregnes efter formlen $\epsilon_j = s_j' \sum_i \delta_i w_{ij}$, der udtrykker fejlen på det skjulte lag gennem fejlen på outputlaget.

Det er af denne grund algoritmen kaldes *backpropagation of errors*. Fejlen på outputtet føres baglæns op gennem netværket fra output mod input.

Det er ikke en tilfældighed at de to lag behandles på samme måde. Lignende regler gælder for helt generelle netværk: vægtændringen for en forbindelse er proportional med fejlen, der hvor den ankommer, multipliceret med aktiviteten, der hvor den udgår fra.

### Træning og generalisation

I praksis er der altid mere end et eksempel. Men en simpel analyse viser, at man blot skal præsentere eksemplerne i systematisk eller tilfældig rækkefølge og anvende ovenstående algoritme på hvert af dem. Igen og igen ændres der på vægtene, medens den gennemsnitlige fejl på alle eksemplerne formindskes jævnt. Til sidst når fejlen et minimum, der kan være nul, men dog sjældent er det. Det er denne proces, som kaldes *træning* af netværket.

Netværkets *generalisationsfejl* bestemmes som den gennemsnitlige fejl på eksempler, der ikke er blevet brugt til træningen. Generalisationsfejlen er normalt større end træningsfejlen. I figur 3 ses træningsforløbet for et klassisk statistisk eksempel, som angår forudsigelse af antallet af solpletter næste år ud fra antallet i de foregående år (der er en berømt 11-års periode).

Det bemærkes i figur 3, at som funktion af træningstiden ser generalisationsfejlen ud til at vokse igen efter et vist punkt. Den optimale træningstid svarende til den mindste generalisationsfejl ligger på omkring 40 epoker (en epoke er en fuldstændig præsentation af træningssættet). Dette er et karakteristisk fænomen, som skyldes, at der er mere plads i netværket end det er nødvendigt for at omfatte de (for os ukendte) regler, der beskriver solpletforekomsterne. Når netværket har lært disse, går det videre og prøver at lære detaljerne i det specielle træningssæt, det har set. I stedet for at optræde som generalist bliver det til en specialist.

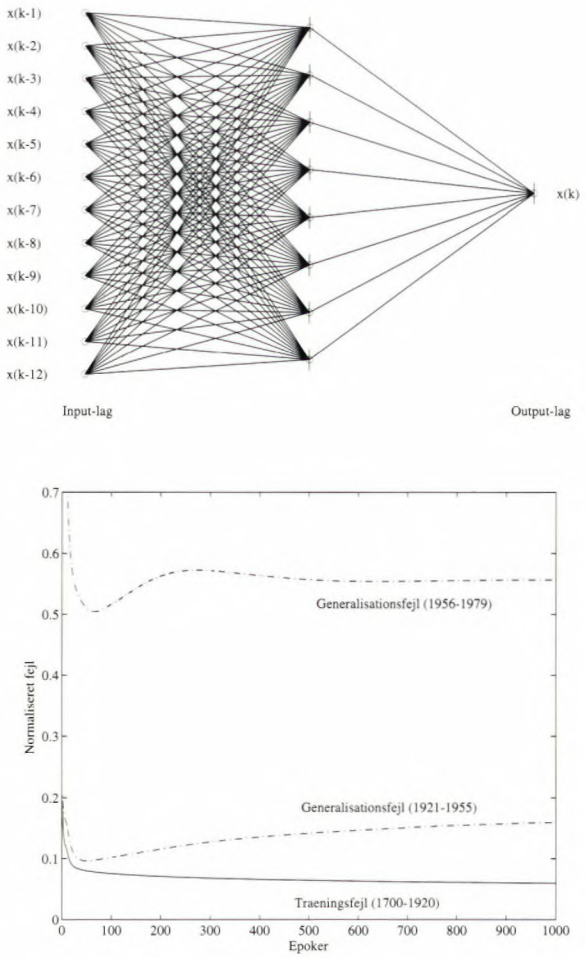
### Kapacitetskontrol

Overtræning er et alvorligt problem, der som nævnt skyldes, at netværkets indre kapacitet er større end krævet af det givne endelige sæt af eksempler. Hvis træningssættet var uendeligt stort ville dette problem ikke forekomme, men det er ikke så interessant, fordi træningen da ville tage uendelig lang tid.

I de otte år, backpropagation har været brugt, er der prøvet mange teknikker for at komme uden om dette problem, men først i de seneste år er der blevet udviklet systematiske og praktisk anvendelige metoder til at kontrollere netværkenes kapacitet. I det følgende gennemgås en række kapacitetskontrollerende foranstaltninger.

**Afhopning:** Den mest naive metode er at stå af, mens legen er god, det vil sige at stoppe træningen før generalisationsfejlen begynder at stige igen. Det er også udmærket, hvis man har masser af eksempler til træning og testning. Men ofte er man i den situation, at man dårligt kan undvære test-eksemplerne, men helst ville bruge alle til rådighed værende eksempler til træningen. Det kan for eksempel dreje sig om at bruge et netværk til at forudsige effektiviteten af en kemisk reaktion, men hvor problemet er, at hvert eksperiment er besværligt og måske tager 24 timer at udføre. I et sådant tilfælde kan man ikke benytte "afhopning" som kapacitetskontrol.

**Vægthenfald:** En anden måde at reducere netværkets kapacitet består i under træningen at lade vægtene henfalde med en konstant faktor for hvert eksempel. Nogle af vægtene ændres under træningen så lidt, at de efterhånden henfalder til nul, medens andre ændres så meget, at de overlever henfaldet. Det største problem med denne metoder er at vælge en fornuftig henfaldskonstant for vægtene. Vælges den for stor, falder alle vægte hurtigt mod nul, og vælges den for lille sker der intet.



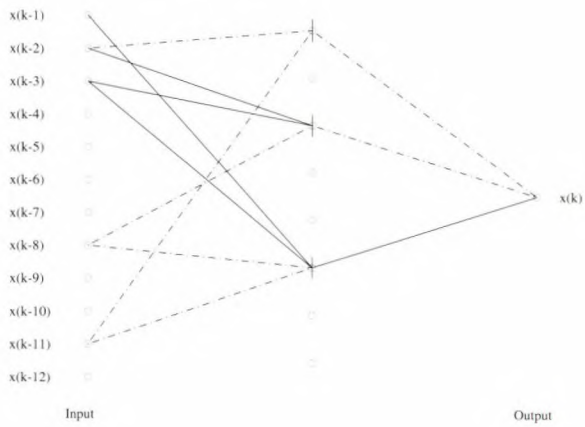

**Figur 3.** Indlæring i et neuralt netværk. Det drejer sig her om et netværk, som lærer at forudsige næste års solpletantal ud fra de foregående 12 års. Øverst ses selve det fuldt forbundne netværk med 12 input og 1 output. Nederst ses træningsforløbet (beregnet af Claus Svarer). Træningseksemplerne er årene fra 1700 til 1920, og generalisationsevnen bestemmes i to perioder, den ene fra 1921 til 1955 og den anden fra 1956 til 1979. Der er en mystisk kvalitativ forskel på de to perioder (*new age?*). På den horisontale akse afbildes antallet af gange træningssættet er blevet præsenteret for netværket. Epokerne er altså et mål for træningstiden.

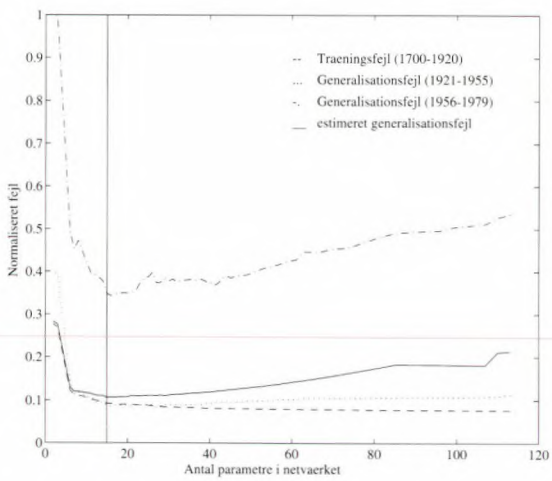
**Beskæring:** Når en vægt bliver nul, kan man lige så vel kappe forbindelsen, altså forandre netværkets arkitektur. Det er også en effektiv måde at reducere kapaciteten på. I stedet for at bruge vægthenfald, kan man gå direkte ind og fjerne forbindelser, der ikke ser ud til at have nogen særlig betydning. For eksempel kan man slette de små vægte.

Det har dog den fare indbygget, at man derved ville slette små men betydningsfulde vægte. I det følgende vil vi beskrive et par metoder til arkitekturoptimering, som ikke har dette problem.

### Optimal Brain Damage (OBD)

Denne metode med det ret makabre navn er udviklet i USA på AT&T Bell Labs til brug i deres system til genkendelse af håndskrevne tal (postnumre). Man beregner simpelthen, hvilke vægte der har mindst indflydelse på fejlen, og sletter dem derefter.




**Figur 4.** Beskæring af netværk. Øverst ses det færdigt beskårne netværk, som har 15 parametre (vægte og tærskler). Nederst ses beskæringsforløbet som funktion af antallet af vægte i netværket (beregnet af Claus Svarer). Dette er i øvrigt verdens mest kompakte repræsentation af solpletserien. Beskæringsprocessen foregår fra højre mod venstre, og det ses, hvorledes træningsfejlen langsomt forøges medens generalisationsfejlen formindskes. Den lodrette streg markerer minimum i generalisationsfejlen.

Efter at have klippet en eller flere vægte, skal netværket genoptrænes og processen fortsætter. Den samlede træningsfejl stiger efterhånden som netværket reduceres, medens generalisationsfejlen falder indtil et vist punkt, hvorefter den begynder at stige igen.

### Optimal Brain Surgeon (OBS)

I en videreudvikling af *Optimal Brain Damage* kan genoptræningen til dels undgås ved at beregne de ændringer, der sker i de andre vægte, når en given vægt nulstilles. Derved kan man forfølge minimummets bevægelse under beskæringen. Det kræver imidlertid en omfattende beregning. Til gengæld slipper man for den tidskrævende, men enkle, genoptræning af netværket. Denne metode er dog så ny, at den endnu ikke har fundet praktiske anvendelser. Det er derfor for tidligt at udtale sig om anvendeligheden.

### Stopkriteriet

Det er som omtalt ovenfor ofte tilfældet, at antallet af træningseksempler er meget begrænset, og at det derfor er "uøkonomisk" at reservere et antal eksempler til måling af netværkets generalisationsevne. Vi er så ikke i stand til at bestemme det optimale punkt, hvor netværket lige netop er beskåret så meget, at det befinder sig i mimimum for generalisationsfejlen.

Et ældre resultat fra konventionel statistisk analyse (udledt af den japanske statistiker H. Akaike) kan bringes i anvendelse til at forudsige generalisationsfejlens minimum. For et netværk med $D$ parametre (vægte og tærskler) trænet med $P$ eksempler vurderes den gennemsnitlige generalisationsfejl per eksempel efter følgende formel

$$E_{\text{generalisation}} = \frac{P+D}{P-D} E_{\text{træning}} \qquad (7)$$

Under beskæringen reduceres antallet af parametre, $D$, jævnt, medens antallet af træningseksempler, $P$, er konstant. Det ses af formlen, at den gennemsnitlige generalisationsfejl og træningsfejl derved nærmer sig hinanden. I figur 4 er Akaike's resultat vist som den fuldt optrukne linie, og det er tydeligt at det forudsagte mimimumspunkt falder sammen med det, der er bestemt ud fra de to sæt af eksempler brugt til testning.

### Grænser for vækst

Den samme formel (7) kan også bruges til at stoppe vildtvoksende netværk. I litteraturen er der nemlig fremkommet et stort antal vækstalgoritmer, som automatisk tillader et netværk at vokse, således at det kan rumme enhver datakompleksitet. Oftest går de ud på at træne et givet netværk og derefter på grundlag af en analyse af de fejl, netværket gør, tilføje en eller flere neuroner, der kan korrigere disse fejl.

På trods af de mange modeller, er der ingen enighed om, hvilken der er bedst. Måske er styret vækst ikke sådan en god ide, fordi filosofien dybest set er forkert. Det kan være, det er ufrugtbart at lade nye neuroner rette de fejl, de gamle laver.

En anden kritik går på, at neuroner simpelthen er for store klumper af parametre at lægge til et netværk.

Det ville være nyttigt med en algoritme, der modsat OBD kunne lægge forbindelser ind, der hvor de var mest nødvendige. En sådan algoritme kendes desværre ikke.

Måske er det mest hensigtmæssigt simpelthen at skabe et overskud af forbindelser for derefter at skære dem bort, der ikke er påkrævede. Når vi fødes, er der for eksempel flere forbindelser til musklerne end nødvendigt, men under brugen af musklerne reduceres dette antal hurtigt. Naturens egne metoder til neural vækst er blevet udviklet gennem en milliard år og berettiger nok til et indgående studium.

*Benny Lautrup* er lektor i teoretisk fysik ved Niels Bohr Institutet og leder af det danske forskningscenter for neurale netværk: CONNECT.

# Biologisk beregning

*John Hertz*

Hvis vi vælger at kalde det, hjernen gør, for beregning, er det klart nok af en temmelig anden form, end den vi kender til i de sædvanlige computere eller i *feedforward* neurale netværk. Så simple modeller kan ikke beskrive systemer med så udstrakt et *feedback*, som hjernen har. I de senere år har neurovidenskaben taget ideer op fra fysikken, for eksempel fra teorien for dynamiske systemer og fra den statistiske mekanik, for at finde nye formuleringer på de neurovidenskabelige problemstillinger.

Neurovidenskabens mål er at karakterisere psykologiske fænomener, som for eksempel sansning, opmærksomhed, hukommelse og bevidsthed, gennem dynamikken i hjernens neurale netværk. I lang tid vægrede neurobiologerne sig mod at studere sådanne problemer på grund af vanskeligheder med at foretage eksperimenter, som kunne kaste et klart lys over dem. I de seneste år har begreber importeret fra fysikken imidlertid gjort det muligt at konstruere et teoretisk underlag for eksperimentel undersøgelse af disse fundamentale problemer. I denne korte artikel vil jeg kort beskrive de vigtigste ideer og nogle eksperimenter til at teste dem med.

## Attraktorer

Hjernen er langt mere kompleks end noget kunstigt neuralt netværk, som vil blive konstrueret inden for en overskuelig fremtid. Ikke desto mindre er den et dynamisk system, som er underkastet den klassiske fysiks love. Desuden er den et såkaldt *dissipativt* system, hvilket vil sige, at den forbruger energi. Vi ved, at den asymptotiske opførsel af sådanne systemer kan beskrives ved *attraktorer* — karakteristiske bevægelsesmønstre som systemet til sidst ender op i.

Der er tre slags attraktorer. Den første type er et *fikspunkt*, som i dette tilfælde ville beskrive et neuronalt aktivitetsmønster, der ikke ændrer sig i tiden. Den anden type er en *grænsecyklus*, hvor aktivitetsmønstret er periodisk i tiden. Endelig findes der *sære attraktorer* hvor aktivitetsmønstret ændrer sig kaotisk i tiden.

For alle tre typer af attraktorer er systemets bevægelse stærkt begrænset i forhold til det fuldstændige tilstandsrum, som det i princippet kunne bevæge sig i. Dette er klart for fikspunkter: attraktoren er et enkelt punkt i et stort rum. Men selv sære attraktorer har en effektiv (fraktal) dimension som er mindre end det fulde tilstandsrum, så at attraktoren har målet nul. At systemets udvikling tiltrækkes mod et så specielt sæt tilstande er en konsekvens af dissipationen og er et meget kraftigt matematisk udsagn om dynamikken.

Walter Freeman fra Berkeley universitetet i Californien har brugt disse resultater fra teorien for dynamiske systemer og har fremsat den formodning, at elementære psykologiske begivenheder kan forbindes med, at hjernen (eller en del af den) falder ind i en attraktor. Ifølge denne tanke, kan enhver sansning, enhver erindring, o.s.v. identificeres med en eller anden attraktor. For eksempel, når et dyr sanser en bestemt duft, så vil de områder som i hjernen forbindes med lugtesansen (det olfaktoriske cortex) falde ind i en attraktor, og sanseoplevelsen af duften er simpelthen identisk med denne proces. Hvis to forskellige kemikalier påvirker duftmodtagecellerne i næsen, vil de