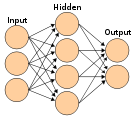
Neurale netværk

Et *neuralt netværk* består af tre eller flere *lag*. Det første lag kaldes *input-laget*. Det sidste lag kaldes *output-laget*. Lagene imellem kaldes *skjulte*.

Figuren til højre illustrerer situationen hvor der er et enkelt skjult lag.

I hvert lag er der et antal *neuroner*, hver illustreret med en cirkel på figuren.

Neuronerne er forbundet med *axoner* – altså pilene på figuren. En pil betyder, at den ene neuron kan påvirker *aktivitetet* i den anden. Størrelsen af denne indflydelse kaldes *vægt*.

Derudover kan der være et konstantled i hvert lag undtagen output-laget. Dette kaldes for en *bias-*enhed.

Den samlede aktivitet i en neuron findes ved at indsætte resultatet af summen af alle axonernes påvirkning i en såkaldt *aktiveringsfunktion*. Vi vil her altid bruge sigmoid-funktionen.

# Matematisk formulering

Betragt et neuralt netværk med lag i alt. Input-laget, der altså er lag nr. 1, består af en observation . Observation , foruden bias-enheden der svarer til og dermed , kan påvirke neuron nr. i første skjulte lag med en vægt . Dvs:

Eller på matrixform:

Her er og . Her angiver antallet af neuroner i andet lag (første skjulte lag). For at få aktiveringen for neuroner i dette lag anvendes sigmoid-funktionen:

Så . Hertil føjes en bias-enhed, så . Næste lag har så på helt analog vis aktiveringen:

Igen tilføjes en bias-enhed osv. Indtil output-laget nås:

# Likelihood-funktion

Vi kigger her kun på et enkelt datasæt , hvor og altså er vektorer. Tolkningen af værdien af det ’te element i er sandsynligheden for at er lig 1. Altså et Bernoulli-eksperiment. Derfor er likelihood-funktionen:

Den tilhørende log-likelihood er:

Denne størrelse, som dybest set er det samme som omkostnings-funktionen - skal minimeres. Derfor ønsker vi at finde de afledede mht. de forskelle vægte i . Dette gøres ved hjælp af den såkaldte *tilbagepropagerings*-algoritme (*back propagation*).

## Afledte af omkostningsfunktionen – indledende forsøg

Vi er interesserede i at aflede efter vægtene i de forskellige lag:

# Tilbagepropagation (back-propagation)

## Fejl i neurale netværk

Betragt igen et neuralt netværk med lag. Vi kigger her igen kun på et enkelt datasæt . Forudsigelse for mulighed i output-laget (lag er aktiveringen , mens den sande værdi er . ”Fejlen” i dette lag er altså:

Alternativt kunne dette skrives:

Eller på vektorform:

Denne fejl *tilbagepropageres* nu til sidste skjulte lag vha. følgende formel:

Her er ’erne inputværdierne for sidst skjulte lag. Da den afledte af sigmoid-funktionen er givet ved: er dette det samme som:

Herefter kan man ”fortsætte bagud” endnu et lag:

Osv. Indtil man har en fejlvektor for alle de skjulte lag. Inputlaget har ikke nogen fejlvektor.

## Partielle afledte – et datasæt

De partielle afledede af omkostningsfunktionen – stadig kun for ét datasæt - er nu givet ved:

## Partielle afledte – alle datasæt

For at finde afledte for alle datasæt skal denne beregning udføres for hver af de enkelte datasæt, summeres (og normaliseres ved at dividere med som sædvanligt). I praksis gøres dette ved at bruge akkumulatorer :

1. Sæt alle
2. Beregn alle for et givent datasæt
3. Brug tilbagepropagation til at beregne
4. Opdater akkumulatoren: Læg til
5. Gentag trin 2-4 for alle datasæt
6. Beregn de partielle afledede som

## Regularisering

Når regularisering inkluderes kommer der et ekstra led på ikke-bias-elementerne:

Her er -leddet blot en smart at undgå regularisering af bias-vægtene. De afledede beregnes nu:

# Teoretisk udledning

## Partielle afledte

Vi kigger igen på tilfældet hvor der kun er ét datasæt . Vi kan tænke på omkostningsfunktionen som en sammensat funktion:

Her er fokus på sammenhængen fra sidste skjulte lag til output-laget – vi tænker på alle andre vægte som værende konstanter. Lad os aflede efter vægtene i det sidste, skjulte lag:

(Bemærk at faktorerne alle er matricer). Hvis man er interesseret i at aflede efter vægtene i næstsidste, skjulte lag må man tænke på omkostningsfunktionen således:

Dermed kan de relevante afledte skrives:

Tidligere skjulte lag vil få tilføjet yderligere faktorer efter sammen mønster. Vi bemærker, at de to første faktorer er ens i alle udtrykkene:

Tilsvarende kan vi definere

Og mere generelt:

Generelt kan de afledte nu skrives:

## Sidste skjulte lag

Lad os beregne de afledte mht. vægte i lag :

Husk at omkostningsfunktionen på vektoriseret for er:

Derfor er:

Aktiveringsfunktionen er sigmoid-funktionen, så:

Så: