

Artificiella Neuronnät

- 1 Artificiella neuronnät
 - Karaktäristiska egenskaper
 - Användningsområden
 - Klassiska exempel
 - Biologisk bakgrund
- 2 Enlagersnät
 - Begränsningar
 - Träning av enlagersnät
- 3 Flerlagersnät
 - Möjliga avbildningar
 - Backprop algoritmen
 - Praktiska problem
- 4 Generalisering

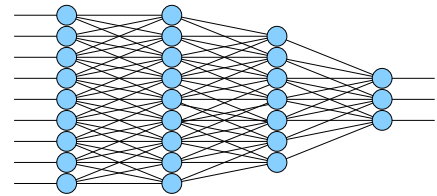
- 1 Artificiella neuronnät
 - Karaktäristiska egenskaper
 - Användningsområden
 - Klassiska exempel
 - Biologisk bakgrund
- 2 Enlagersnät
 - Begränsningar
 - Träning av enlagersnät
- 3 Flerlagersnät
 - Möjliga avbildningar
 - Backprop algoritmen
 - Praktiska problem
- 4 Generalisering

Artificiella neuronnät (ANN)

- Inspirerade av hur nervsystemet fungerar
- Parallell bearbetning

Vi begränsar oss här till **en** typ av ANN:

Framåtkopplade nät med lagerstruktur



Användningsområden

Fungerar i princip som en generell "Lärande låda"!

Klassificering



Funktionsapproximering



Flerdimensionell avbildning

Klassiska exempel

ALVINN

System för att styra en bil

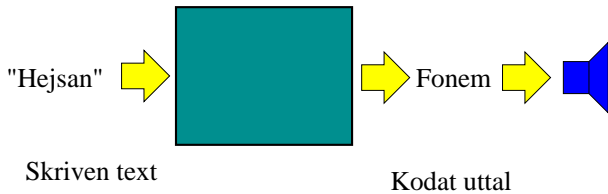


Tränas med riktiga förarens beteende som träningsexempel

Klassiska exempel

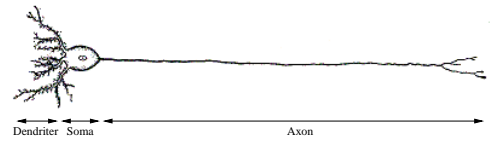
NetTalk

Program för talsyntes



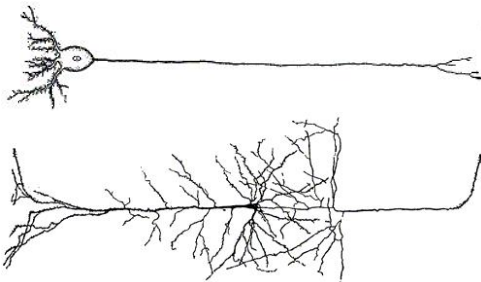
Tränas med stor databas över uttalad text

Hur fungerar riktiga nervceller?



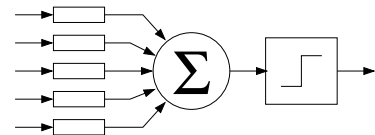
- **Dendriter**
Passiv mottagning av (kemiska) signaler
- **Soma (cellkropp)**
Summering, tröskling
- **Axon**
Aktiva pulser sänds till andra celler

Nervceller kan se väldigt olika ut



ANN-karikatyren

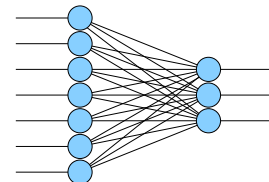
(en förenklad bild av informationsbehandlingen)



- Viktade insignaler
- Summering
- Trösklad utsignal

- 1 Artificiella neuronnät
 - Karaktäristiska egenskaper
 - Användningsområden
 - Klassiska exempel
 - Biologisk bakgrund
- 2 Enlagersnät
 - Begränsningar
 - Träning av enlagersnät
- 3 Flerlagersnät
 - Möjliga avbildningar
 - Backprop algoritmen
 - Praktiska problem
- 4 Generalisering

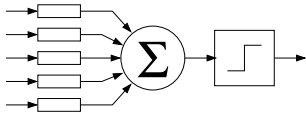
Vad menas med ett *Enlagersnät*?



Varje cell fungerar oberoende av de andra!

Det räcker att förstå vad *en* cell kan beräkna

Vad kan en enstaka "cell" beräkna?

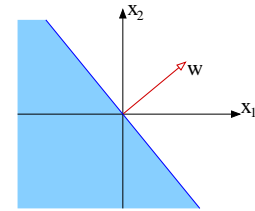


- \vec{x} Indata i vektorform
- \vec{w} Vikterna i vektorform
- o Utsignalen

$$o = \text{sign} \left(\sum_i x_i w_i \right)$$

$$o = \text{sign} \left(\sum_i x_i w_i \right)$$

Geometrisk tolkning



Separerande hyperplan
Linjär separerbarhet

Inläring i ANN

Vad innebär inläring?

Nätets struktur är normalt fix

Inläring innebär att hitta de **bästa vikterna** w_i

Två bra algoritmer för enlagersnät:

- Perceptroninläring
- Deltaregeln

Perceptroninläring

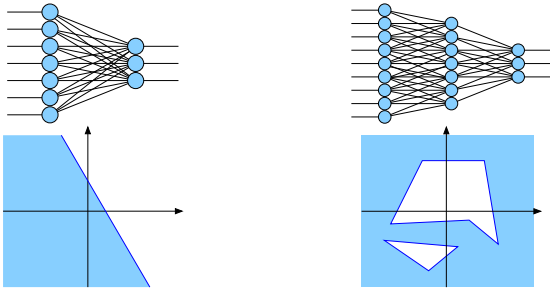
- Inkrementell inläring
- Vikterna ändras bara vid felklassificering
- Uppdateringsregel: $w_i \leftarrow w_i + \eta(t - o)x_i$
- Konvergerar alltid om klassningsproblemet är lösbart

Deltaregeln (LMS-regeln)

- Inkrementell inläring
- Vikterna ändras alltid
- $w_i \leftarrow w_i + \eta(t - \vec{w}^T \vec{x})x_i$
- Konvergerar endast i statistisk mening
- Hittar en optimal lösning även om problemet inte är exakt lösbart

- 1 Artificiella neuronnät
 - Karaktäristiska egenskaper
 - Användningsområden
 - Klassiska exempel
 - Biologisk bakgrund
- 2 Enlagersnät
 - Begränsningar
 - Träning av enlagersnät
- 3 Flerlagersnät
 - Möjliga avbildningar
 - Backprop algoritmen
 - Praktiska problem
- 4 Generalisering

Vad är poängen med att ha flera lager?



Ett tvålagersnät kan implementera **godtyckliga beslutsytor**
...förutsatt att man har *tillräckligt många gömda enheter*

Blir det ännu bättre med ännu fler lager?

- Två lager kan beskriva **alla** klassificeringar
- Två lager kan approximera **alla** "snälla" funktioner
- Tre lager kan ibland göra samma sak effektivare
- Fler än tre lager används mycket sällan

Hur tränar man ett flerlagersnät?

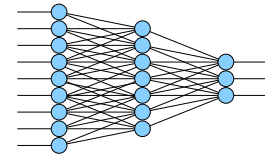
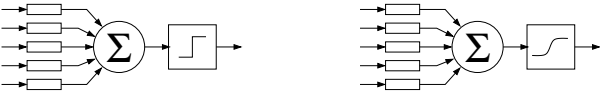
Varken perceptroninläring eller deltaregeln kan användas

Fundamentalt problem:

När nätet svarar fel vet man inte i vilken riktning viktorna ska ändras för att det ska bli bättre

Avgörande trick:

Använd trösklingsliknande kontinuerliga funktioner



Grundidé:

Minimera felet (E) som en funktion av **alla** vikter (\vec{w})

- 1 Beräkna den riktning i viktrummet dit felet ökar mest:
 $\text{grad}_{\vec{w}}(E)$
- 2 Ändra viktorna i motsatt riktning

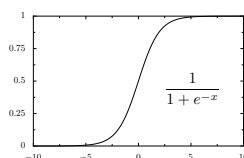
$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$$

Normalt utnyttjar man felet för ett exempel i taget

$$E = \frac{1}{2} \sum_{k \in \text{Out}} (t_k - o_k)^2$$

Som trösklingsliknande funktion används ofta

$$\rho(y) = \frac{1}{1 + e^{-y}}$$



Gradienten kan uttryckas som en funktion av ett *lokalt generaliserat fel* δ

$$\frac{\partial E}{\partial w_{ji}} = -\delta_i x_j \quad w_{ji} \leftarrow w_{ji} + \eta \delta_i x_j$$

Utlagret:

$$\delta_k = o_k \cdot (1 - o_k) \cdot (t_k - o_k)$$

Gömda lager:

$$\delta_h = o_h \cdot (1 - o_h) \cdot \sum_{k \in \text{Out}} w_{kh} \delta_k$$

Felet δ propagerar bakåt genom lagren
Error backpropagation (BackProp)

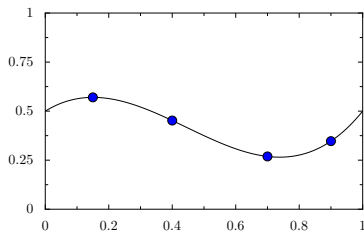
Att tänka på när man använder BackProp

- Långsam
Kräver normalt tusentals genomgångar av datamängden
- Gradientföljning
Riskerar att fastna i lokala minima
- Många parametrar
 - Steglängden η
 - Antal lager
 - Antal gömda enheter
 - In- och utrepresentationen
 - Initiala viktvärden

- 1 Artificiella neuronnät
 - Karakteristiska egenskaper
 - Användningsområden
 - Klassiska exempel
 - Biologisk bakgrund
- 2 Enlagersnät
 - Begränsningar
 - Träning av enlagersnät
- 3 Flerlagersnät
 - Möjliga avbildningar
 - Backprop algoritmen
 - Praktiska problem
- 4 Generalisering

Generalisering

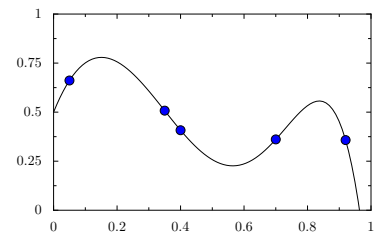
Näten *interpolerar* normalt mjukt mellan träningsdatapunkterna



Ger oftast bra generalisering

Risk för överinläring!

Om nätet har för många frihetsgrader (vikter) är risken större att inläringen hittar en "konstig" lösning



Genom att begränsa antalet gömda noder får man bättre generalisering

