Faculteit der Exacte Wetenschappen

(Duidelijk en met blokletters invullen)

1 2 3 4 5
15 | 10 10 | 5 10 10 | 10 | 10 15 | 10

Cijfer:

Naam en voorletters:

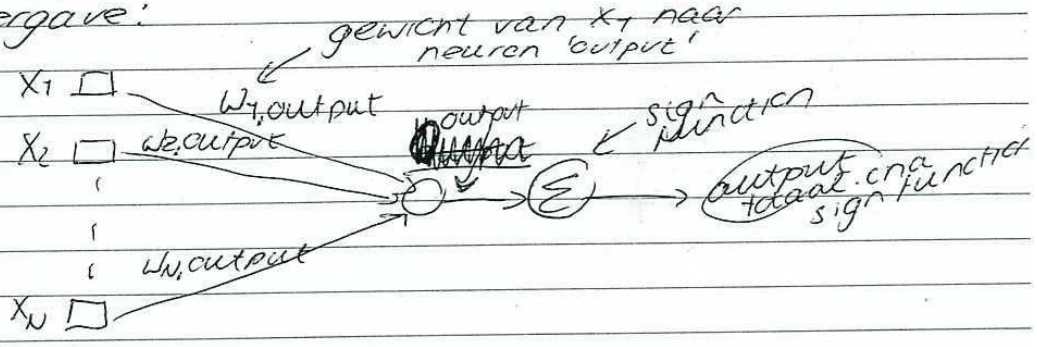
Vak: Neurale netwerken **Studentnummer:**

vrije Universiteit amsterdam

Datum: 26-05-03 **Jaar van 1e inschrijving:** 99 **Studierichting:** BWL

1. 15

- Perceptron architecture: architectuur van een perceptron bestaat uit één enkele laag (single layer). Dit houdt in dat de input neuronen rechtstreeks verbonden zijn met de output neuron. Een perceptron heeft maar één output neuron. Grafische weergave:



- Neuron model (information processing unit): in de output neuron wordt de sign functie gebruikt. Deze is als volgt gedefinieerd:

$$\varphi(v_j) = \begin{cases} 1 & \text{als } v_j \geq 0 \\ -1 & \text{als } v_j < 0 \end{cases}$$

$$v_j = \sum_i w_{ij} x_i$$

- Learning algoritm:
Voor de perceptron wordt het fixed increment learning algoritm gebruikt:

$n = 1$
initialize $w(n)$ random.
while (there are missclassified examples)
  select a missclassified example $(x(n), d(n))$
  $w(n+1) = w(n) + \eta x(n) d(n)$
  $n = n+1$
end-while.

Hierbij is N het de dimensievector. n het aantal

aanpassingen dat je doet op de gewichtsvector
η de learning rate en $x(n)$ het training
example dat je voor de n^e iteratie gebruikt
en $d(n)$ de ~~data~~ (correcte) classificatie
van dit n^e training example.

- ~~Perceptron kan niet worden toegepast~~

rvolg vraag 7. ~~separable~~ inseparable classes

- Perceptron kan ~~niet~~ worden toegepast op
non-linearly ~~training examples~~ omdat
perceptron alleen kan classificeren d.m.v.
één hyperplane (rechte lijn / vlak) als
examples niet linear separabel zijn, zijn
er dus bijvoorbeeld meerdere lijnen / vlakken nodig
en dus kan de perceptron dit probleem
niet aan.

* Als examples niet linear separabel zijn,
zijn er meerdere lijnen / vlakken
nodig om de examples te scheiden en
dit perceptron kan het slechts met
één hyperplane.

2 • ~~Supervised learning: je input bestaat uit gelabeled input~~

2 • Supervised learning: je data bestaat uit gelabeled inputs, desired outputs). Dit houdt in dat je bij het trainen van je netwerk dus niet alleen de beschikking hebt over labeled inputs (dus de input voor je netwerk) maar je weet ook wat het netwerk bij een bepaalde input als output (=desired output) moet geven. Deze desired outputs gebruik je ook bij het trainen van je netwerk. Voorbeelden van supervised learning zijn perceptron, adaline, feed forward neural networks en RBF ~~radial basis functions (FFNN).~~

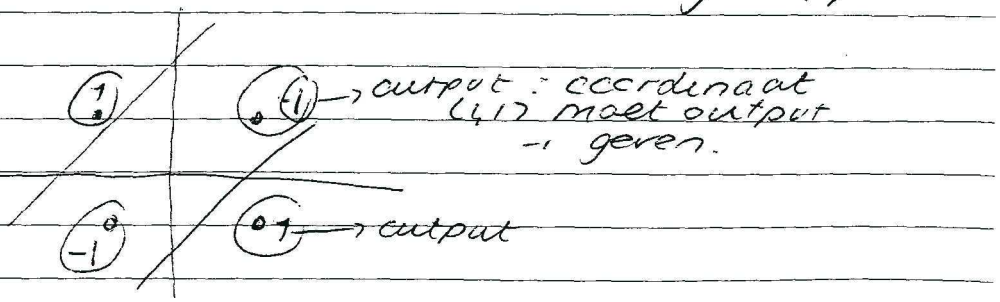
Unsupervised learning: Bij unsupervised learning heb je alleen beschikking over inputs en weet je dus niet welke output er bij een dergelijke ~~(soort)~~ input hoort. Je gaat het netwerk trainen ~~op basis van~~ met behulp van alleen je inputs. (je kunt bijvoorbeeld structuur in je inputs ontdekken). Voorbeelden van unsupervised learning zijn SOM en hopfield.

• a FFNN that solves the XOR problem.

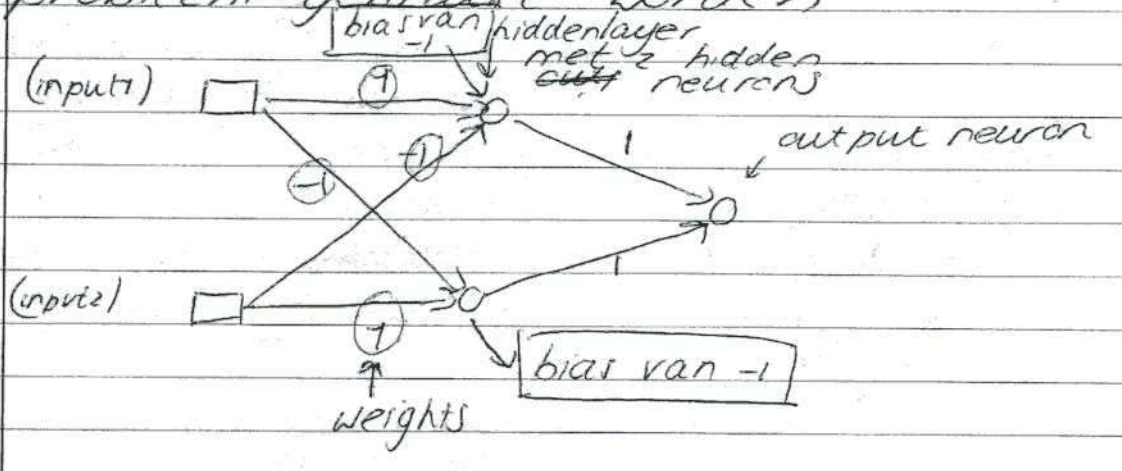
In dit geval moet gelden:

Input	gewenste output
(1,1)	-1
(-1,-1)	-1
(1,-1)	1
(-1,1)	1

We moeten dus een FFNN vinden die deze inputs als volgt scheidt: (d.m.v. 2 rechte lijnen)



~~Een FFNN die het XOR probleem oplost~~
Een FFNN die het XOR probleem oplost bestaat uit: twee input neuronen, twee hidden neuronen en 1 output neuron. Voor het neuron model van de hidden neuronen en de output neuron geldt dat de sign activation function wordt gebruikt. Met volgende netwerk kan voor het XOR probleem gebruikt worden



Hierbij ga ik ervan uit dat de sign activation function als volgt werkt:

$$\varphi(v_j) = \begin{cases} 1 & \text{als } v_j \geq 0 \\ -1 & \text{als } v_j < 0 \end{cases}$$

Met $v_j = \sum x_i \cdot w_{ij}$ $\quad i=1,2 \quad j=1,2$ (in dit geval)

Voorbeeld (7, -1)

Output eerste hidden neuron =
sign(1·1 + -1·-1 + (-1)) = sign(1) = 1
(-1) = bias

Output tweede hidden neuron =
sign(1·-1 + -1·1 + (-1)) = sign(-3) = -1.
(-1) = bias

Output output neuron =
sign(1·1 + -1·-1) = sign(2) = 1. (klopt!)

Faculteit der Exacte Wetenschappen

(Duidelijk en met blokletters invullen)

vrije Universiteit amsterdam

Naam en voorletters:

Cijfer:

Vak: Neurale Netwerken **Studentnummer:**

Datum: 26-05-03 **Jaar van 1e inschrijving:** 99 **Studierichting:**

2 J. 5

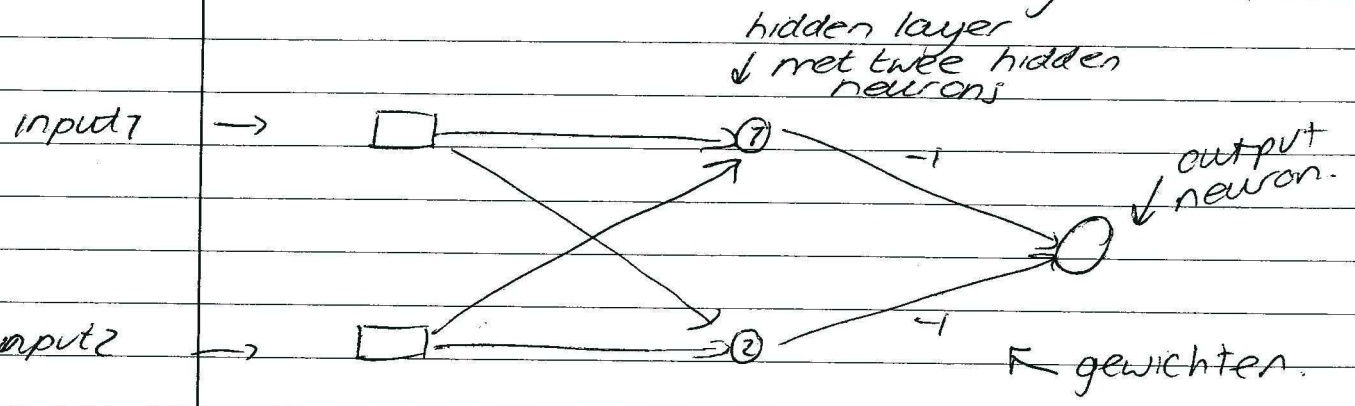
° Een radial basis function (RBF) is een functie waarvan de output afhangt van (een niet stijgende ~~functie~~ functie) afstand tussen een input vector en ~~een~~ stored vectors.
Bij een radial basis function wordt dus de output van een input vector berekend a.d.h.v. 'stored vectors' en de gewichten, ~~van deze stored vectors~~ behorend bij deze stored vectors.

de afstand van deze input tot stored vectors

° Een RBF netwerk bestaat ~~dus~~ uit M1 hidden neurons die ieder een 'eigen' RBF activation function hebben en een 'eigen' center t_i ($i = 1, \ldots, M1$). De input van de activation function van een bepaald hidden neuron is (in ieder geval) $\|x - t_i\|$ => de afstand tussen het input examplexen ~~de~~ de center t_i die bij die ~~neuron~~ hidden neuron hoort. Er zijn verschillende mogelijkheden voor RBF activation functions. In bepaalde gevallen wordt de gaussian RBF activation function gebruikt, dan hebben we ook de spreiding (6) nodig van de centers

10

10 • Een RBF netwerk dat het XOR probleem oplost bestaat uit 2 input neuronen, 2 hidden neuronen ~~met RBF activation functions~~ (met RBF gaussian activation functions.) Het netwerk ziet er als volgt uit:



Het center dat bij hidden neuron 1 hoort is $t_1 = (1,1)$. Het center dat bij hidden neuron 2 hoort is $t_2 = (0,0)$. De RBF gaussian activation die bij hidden neuron 1 hoort is $\varphi_1 = e^{-\|x - t_1\|^2}$ met $t_1 = (1,1)$. De RBF gaussian activation function die bij hidden neuron 2 hoort is $\varphi_2 = e^{-\|x - t_2\|^2}$ met $t_2 = (0,0)$

De output van het netwerk geeft dat de input x tot class 1 behoort als $-e^{-\|x - t_1\|^2} - e^{-\|x - t_2\|^2} + 1 > 0$, anders behoort de input tot class -1.

4a

als ...

$t_1 = (1,1)$
$t_2 = (0,0)$

$\begin{pmatrix}1\\1\end{pmatrix}\ \begin{pmatrix}1\\-1\end{pmatrix}$ 1 $\quad -1 - e^{-2} \quad -e^{-4} - e^{-2}$

$\begin{pmatrix}0\\0\end{pmatrix}\ \begin{pmatrix}1\\-1\end{pmatrix}$ 2. $\quad -e^{-2} - 1$

$\begin{pmatrix}1\\1\end{pmatrix}\ \begin{pmatrix}-1\\1\end{pmatrix}$ 1

$\begin{pmatrix}0\\0\end{pmatrix}\ \begin{pmatrix}-1\\1\end{pmatrix} = t_2$

4. Het k-means algoritm:

10 - initialiseer de gewichten w random. Elk gewicht representeert een cluster.
- Kijk voor iedere input bij welke cluster deze hoort door naar de afstand tussen deze input en de gewichten te kijken. Kies voor het gewicht met de minimale afstand (tussen dit gewicht en het input example)

$$c(x) = \underset{j}{\operatorname{argmax}} \|X - W_j(n)\|$$

- Bereken de gewichten nu als volgt:

$$w_j(n+1) = \sum x_j / n_j$$

voor alle x die tot het cluster van wj behoren.

n_j = aantal x (example) in cluster van wj

- Voeg n op: $n = n+1$, en ga door totdat de gewichten niet tot bijna niet meer veranderen

5. • De gewichten van het Hopfield netwerk

10 worden als volgt berekend:

(N = input dimensie)
(M = aantal 'stored memorie vectors')

$$w_{ji} = \frac{1}{M} \sum_{i=1}^{N} \sum_{j=1}^{N} f_{vi} f_{vj} \text{ met } i \neq j \text{ als } i \neq j.$$

$$w_{ji} = \frac{1}{M} \sum_{v=1}^{M} f_{vi} f_{vj} \text{ als } i \neq j. \quad 0 \text{ als } i = j$$

f_{vi} = waarde van i^e component van v^e stored memorie vector

Dit houdt in dat als componenten i en j in veel stored memories bijvoorbeeld positief gecorreleerd zijn dat w_{ji} een groot getal is.
(waarschijnlijk)

Dus: w_{ji} hangt af van de correlatie/afhankelijkheid tussen componenten i en j in stored memorie vectors.

10 • Compare Hopfield en BSB networks:

* Hopfield en BSB networks zijn beide vormen van associated memories networks. De networks komen ~~een~~ op verschillende punten overeen:
- bij beide netwerken daalt, iedere keer bij een verandering van de neuron states, de energie functie.
- Dit houdt in dat de energie dus daalt, en uiteindelijk, als de attractor state bereikt is, de energie tot een minimum is gedaald.
- Beide netwerken werken met attractor states.
- Beide maken gebruik van feedback. Verschil: in een hopfield network is er geen sprake van self feedback, in een BSB wel.
- Beide netwerken maken gebruik van 'Hebb's postulate of learning'.

Verdere verschillen zitten m.n. in de toepassingen:
- BSB staat bekend om zijn clustervaardigheden zoals concept formation

- Hopfield networks zijn meer gericht op 'content addressable memories' en optimalisatie.

Onder verschil: Bij Hopfield wordt de state vector
x asynchroon (d.w.z één voor één (een component vd state vector en random))
aangepast, bij BSB wordt in iedere iteratie
de gehele state vector aangepast.)