Faculteit der Exacte Wetenschappen

1/2

(Duidelijk en met blokletters invullen)

100 → 105

Cijfer: 10

Naam en voorletters:

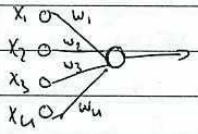
Vak: Neural Networks **Studentnummer:**

Universiteit *amsterdam* **Datum:** 2-6-2005 **Jaar van 1e inschrijving:** 2001 **Studierichting:** BWI

1.(i). The architecture of a perceptron is single-layer feed forward. The neuron uses a non-linear function, it uses a sign-function, which is the following:

$$u = \begin{cases} +1 & \text{if } v \geq c \\ -1 & \text{if } v < c. \end{cases}$$

where c is chosen. A single-layer feedforward is the following:



where x_i are input and w_i are the weights to the neuron

The learning algorithm is the following:

$n=1$; $w(n)$ random

while (there are misclassified examples)

select a misclassified augmented example $(x(n), d(n))$.

$w(n+1) = w(n) + \eta\, d(n)\, x(n)$

$n = n+1$

end-while

Here, $d(n)$ is de desired output.

1(ii). The differences can be are given in the following table.

- Perceptron : minimize the total of misclassified examples.
  Adaline : minimize the total error.
- Perceptron uses a non-linear function, like the sgn-function.
  Adaline uses a linear function. (Activation functions)
- Perceptron can be used to for classification tasks, while adaline can also be used for regression.

1(iii) An example of a boolean function that cannot be learned by a perceptron is XOR. (exclusive or).

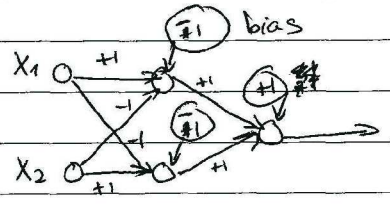
2(i). ~~An unsupervised learn~~

10 ✓ With supervised learning, there is input with desired output. By with unsupervised learning, there are only different realizations of the input.

So, an example of a supervised learning task can be classification of people into categories where the desired category is known. For example, credit-worthy classification.

An example of unsupervised learning task can be clustering, where the goal is to find a relation between the inputs (instead of classifying them).

10 ✓ 2(ii). The architecture and the weights of a FFNN for the XOR function, is the following:



↑ input layer ↑ hidden layer

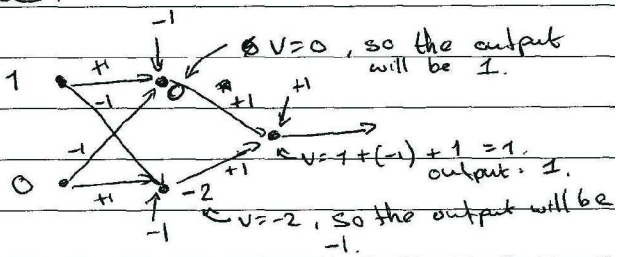
In hidden layer, the sign function is used. Also in the output, the sign function is used. ~~Bias is always~~ Sign function is given by:

$$\begin{cases} 1 & \text{if } v \geq 0 \\ -1 & \text{if } v < 0. \end{cases}$$

Lets ~~the~~ evaluate ~~this~~ the above:

x_1	x_2	output
1	0	1
0	1	1
0	0	0
1	1	0

← schame:

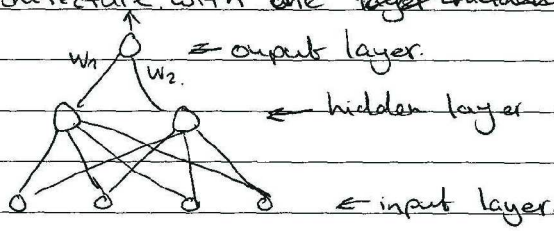


This checks that the architecture and weights do the job. Perhaps, there are more possibilities, but this one does the job correctly.

N.B. the circles denote the bias for the neuron. In the hidden layer, the bias is -1, in the last layer the bias is +1.

10 V

3(c). The architecture of a RBF net is a multi-layer feed forward architecture with one ~~layer~~ hidden layer.



The hidden layer uses an RBF activation function where the distance is measured between the input and the centers of the circles. Each hidden neuron denotes a circle / ball.

So, the ~~output~~ of input that is received by ~~the~~ the output layer is: $w_1 \varphi_1(\|x-t_1\|) + \ldots + w_{m_1} \varphi_{m_1}(\|x-t_{m_1}\|)$

Where, t_i denotes the center of a circle. We can use the following function for φ_i : $e^{-\|x-t_i\|^2}$. So, ~~the output~~ this will be a high value if x is close to the center and has a lower value when x differs much from the center.

There are 3 ~~things~~ parameters that have to be chosen / to be learned:

1) the centers of the circles / balls
2) the spread of the circles / balls.
3) the weights from the hidden layer to the output layer.

10 V

4(c). ~~Both~~ K-means and SOM try to cluster. But with SOM, the neurons that are close to each other are placed together in a grid. So, SOM updates not only one single ~~neuron~~ weight, but also the weights of neighbors of the winning neuron. In the first couple of steps, this neighborhood is large (and contains (almost) every neuron), but when the learning algorithm passes in time, the neighborhood shrinks. This proces makes that neurons that are close to each other are placed near each other in a grid.

Here follows the learning algorithm of SOM:

$n=0$; $w(n)$ random.

draw x from examples

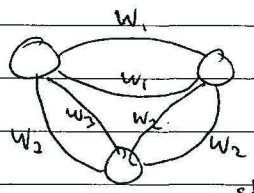
determine winning node: $i(x) = \arg\min_j \|x - w_j(n)\|$

Then update the weights according to: $w_j(n+1) = w_j(n) + \eta(n)\, h_{i(x)}(d_{i(x)j}) \cdot (-w_j(n) + x(n))$

where $h_i = e^{-\frac{d_{ij}^2}{2\sigma^2}}$.

5 V 5(i). Number of neuron: this number is equal to the number of components of a fundamental memory. For example, if a fundamental memory is like "(1, 0, 1, 1)", then there are 4 components and so the number of neurons is also 4.

Architecture: ~~It is an architecture with links to all~~ The architecture looks like:



NN execution

~~Weights computation~~: the ~~weights~~ states of the neurons change according to:

- select a neuron at random, say neuron i.
- for that neuron i, compute the output; i.e. $\sum_j w_{ji} x_i$, where w_{ji} denotes the weight from neuron i to neuron j.

If the output is >0, then the ~~weigh~~ state of the neuron will be $+1$; if the output is <0, the the state of the neuron will be -1; if the output $= 0$, the state will not be changed.

When $x(n) = x(n+1)$, the network is called stable and the state of the network is the vector of the output of each neuron.

Weights computation: Let μ be the fundamental memory. And let $f_{\mu,i}$ be the i-th component of μ.

Then:

$$w_{ji} = \begin{cases} \frac{1}{M} \sum_{\mu=1}^{M} f_{\mu,i} f_{\mu,j} & \text{if } i \neq j. \\ 0 & \text{if } i = j \end{cases}$$

where M is the amount of fundamental memories.

15 6(i) The perceptron gives a line (or hyperplane in higher dimensions) that separates the two classes. In SVM, there are x_i's with α_i's bigger than zero, which are then called "support vectors".

With SVM, the task is to minimize $\|W\|^2$ under some restrictions, which are:

$w^T x + b \geq +1$ if output $= +1$.

$w^T x + b \leq -1$ if output $= -1$.

$\alpha_i \geq 0 \quad \forall i$ (for all i)

$\sum_N \alpha_i y_i = 0$.

Faculteit der Exacte Wetenschappen 8 2/2

(Duidelijk en met blokletters invullen)

Cijfer:

Naam en voorletters:

Vak: Neural Networks **Studentnummer:**

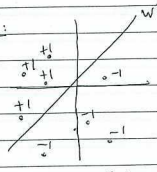
Universiteit *amsterdam* **Datum:** 26-8-05 **Jaar van 1e inschrijving:** 2001 **Studierichting:** BMI

Then w is given by:

$$w = \sum_{i=1}^{N} y_i \alpha_i x_i$$

Let's make a picture as comparison:

perceptron:



SVM: