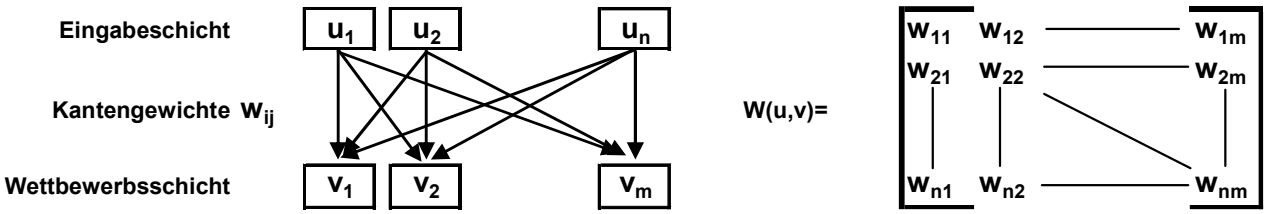
- **Wettbewerbslernen**

Folgend sei für Klassifikationsprobleme der Ansatz eines neuronalen Netzes verwendet:



Zur Lösung von Klassifikationsproblemen können Netze mit einer Wettbewerbsschicht herangezogen werden, wie dies im Beispiel oben für n Eingabeknoten, zur Aufnahme von Merkmalen von Daten und m Wettbewerbsknoten für die eigentliche Klassifikation dargestellt ist.

Wird dem Netz ein Eingabevektor u präsentiert, wird anhand der Ausgabeaktivierung v mit $u^TW = v$ entschieden welchem Wettbewerbsknoten das Eingabemuster (primär) zugeordnet wird. Dies kann beispielsweise über die Funktion $\max(v_1,v_2, ..,v_m)$ geschehen und was wie man leicht einsieht identisch mit dem Problem der Minimierung des Abstandes $||u-w_j||$ über alle Spaltenvektoren w_j, j=1..m, ist[1].

Das Lernen von geeigneten Gewichten eines Netzes unterscheidet man grundsätzlich in überwachtes und unüberwachtes Lernen. Beim überwachten Lernen ist die richtige Ausgabe des Netzes bekannt, so dass anhand einer Fehlerminimierung die geeigneten Gewichte ermittelt werden können.

Beim unüberwachten Lernen fehlt die Information einer Vergleichsausgabe, weshalb das Erlernen der Gewichtsmatrix W aus Strukturerwartungen heraus erfolgt und wobei sich für Klassifikationsprobleme Ähnlichkeitsmaße anbieten.

Ein übliches Verfahren stellt die wiederholte Darbietung von Trainingsdaten in Epochen mit konstanter oder abnehmender Lernrate $\sigma \in (0,1)$ beispielsweise wie folgt dar[2]:

$$w_{j,neu} = \frac{w_{j,alt} + \sigma(u - w_{j,alt})}{\left\| w_{j,alt} + \sigma(u - w_{j,alt}) \right\|} \overset{\sigma=0,5}{=} \frac{w_{j,alt} + u}{\left\| w_{j,alt} + u \right\|}$$

D.h. entsprechend der Lernrate der Trainingsepoche werden die Gewichte sukzessive in die Richtung des gesehenen Eingabevektors verschoben, mit σ=0,5 wird als geeigneter Startlernrate so der Winkel zwischen den Spaltenvektoren der Gewichtsmatrix und dem Eingabevektor gerade halbiert. Die Normierung über die Norm soll geometrische Vergleichbarkeit gewährleisten, die zunächst eine andere Bedeutung als die Normierung von Merkmalsvektoren für deren Vergleichbarkeit in der Addition von Abständen beispielsweise in der Fuzzy-C-Means Clusterung[3] hat.

---

[1] Vgl etwa auch W.König: "Taschenbuch der Wirtschaftsinformatik ....", S. 910.
[2] Vgl Nauck, Klawonn, Kruse: "Neuronale Netze", vieweg 1994 95ff.
In den Beispielen hier wurde die Lernrate konstant gehalten.
[3] Vgl etwa die Ausführungen in http://www.rankingweb.de/Skripte/Software.pdf

Initialisieren wir die Gewichtsmatrix mit der Einheitsmatrix $W_{3x3} = I_{3x3}$ und betrachten wir Trainingsdaten:

u_1	1	0	2
u_2	1	1	2
u_3	0	1	1
u_4	0	0	1
Bandbreite:	1	1	1

so werden alle Eingabevektoren dem Ausgabeneuron v_3 zugeordnet, was aus der höheren numerischen Wertigkeit des Merkmals 3 der Eingabevektoren resultiert. Die Unterschiede in den übrigen Komponenten, werden von der höheren Wertigkeit des Merkmals 3 der Eingabevektoren in der Bedeutung vollkommen überdeckt.

Eingabe		Init W	Siegerneuron u_1,u_2,u_3,u_4	u_4,u_3,u_2,u_1	
u_1	W_1		3	3	
u_2		1 0 0	3	3	
u_3		0 1 0	3	2	
u_4		0 0 1	3	3	
u_1	W_2		1		
u_2		1 0 1	3		
u_3		0 1 1	2		
u_4		1 1 1	1		
			σ=0,5:	σ=0,2:	σ=0,05:
u_1	W_3		1	1	1
u_2		1 0 1	3	3	1
u_3		0 1 0	3	2	2
u_4		1 0 1	1	1	1

Erst wenn die initialisierende Gewichtsmatrix auch die übrigen Komponenten stärker gewichtet, werden - zumeist schon nach der ersten Lernepoche - die Eingabevektoren differenzierter erfasst. Die Lernrate stellt hierbei offensichtlich einen wesentlichen Faktor dar, wie aber auch die Reihenfolge der Darbietung.

Während die Vorabnormierung der Eingabevektoren eine Invarianz bezüglich skalarer Multiplikation aufzeigt, ergeben sich bei eher unüblicher Vorabnormierung der Design-Spaltenvektoren etwas andere aber nicht weniger vielfältige Ergebnisse.

Die Beispieldatei http://www.rankingweb.de/WettbewerbsLernen.xlsm ermöglicht das darbieten der Trainingsdaten mit den angegebenen Initialisierungsmatrizen und mit unterschiedlichen Lernraten sowie auch Reihenfolgen.

Es zeigt sich, dass wesentlich der Unterschied von Wertigkeiten für Konvergenz Sorge trägt und weshalb die empirisch künstlich wirken könnende Normierung von Vektoren sowohl mittels der Euklid-Norm sowie auch linear Wert erhaltend ermöglicht ist um weitere Lösungen aufzeigen zu können.

Analoger Python-Quellcode:

```
import numpy as np
from numpy import linalg as LA

#Vorgabe der Eingabe Vektoren
U = np.array(
 [[1, 0, 2],
 [1, 1, 2],
 [0, 1, 1],
 [0, 0, 1]])
#Und deren Anzahl
U_n = len(U)
#Festlegung der Startmatrix
W = np.array(
 [[1, 0, 1],
 [0, 1, 0],
 [1, 0, 1]])

Lernrate=0.5
Lfaktor=1      # für abnehmende Lernraten <1 wählen
Darbietungen=20

W=W.astype('float64')
for d in range(0, Darbietungen, 1):
    print("Darbietung: ", d+1, " _____" )
    for i in range(0, U_n, 1):
        V=np.array(np.mat(U[i])*np.mat(W))
        j=np.argmax(V)
        hilf=np.array(W[:,j]+Lernrate*Lfaktor*(U[i,:]-W[:,j]))
        euklid=LA.norm(np.mat(hilf), 'fro')
        Wj= np.array(hilf/euklid)
        for k in range(0, 3, 1):
            W.itemset((k,j),Wj.item(k))
        print("Siegerneuron: ", j+1, " zum Eingabevektor", U[i])
```