

# Künstliches binäres Neuron

G.Döben-Henisch  
Fachbereich Informatik und Ingenieurwissenschaften  
FH Frankfurt am Main  
University of Applied Sciences  
D-60318 Frankfurt am Main  
Germany  
Email: doeben\_at\_fb2.fh-frankfurt.de

5. April 2009

## Zusammenfassung

This is a simple example of an artificial neuron with a binary activation function. To read more about this have a look to the online text [www.uffmm.org/anns.html](http://www.uffmm.org/anns.html).

# 1 Anforderungen

Das Modell soll das Verhalten eines einzelnen binären Neurons darstellen.

# 2 Systembeschreibung

## 2.1 Input-Output

Das Neuron hat zwei Eingänge  $IN1$ ,  $IN2$  für *Aktivierungswerte* von anderen Neuronen, zwei Eingänge  $W1$ ,  $W2$  für *Gewichtswerte*, sowie einen Eingang für einen *Schwellwert*  $THETA$ . Dazu einen *Ausgang*  $OUT$ . Die Eingänge  $IN1$ ,  $IN2$ ,  $W1$ ,  $W2$  sowie der Ausgang können den Wert '1' oder '0' haben, der Schwellwert ist eine reelle Zahl  $THETA < 2$ .

## 2.2 Systemfunktion

Die Systemfunktion  $f_{nnbin}$  des binären Neurons bildet die Eingänge auf den Ausgang ab:

$$f_{nnbin} : IN1 \times W1 \times IN2 \times W2 \times THETA \mapsto OUT \quad (1)$$

Die Systemfunktion setzt sich hier aus zwei Teilfunktionen zusammen:

$$f_{nnbin} = net_{nnbin} \otimes act_{nnbin} \quad (2)$$

$$net_{nnbin} = \sum_{i=1}^2 W_i IN_i \quad (3)$$

$$act_{nnbin} = \begin{cases} 1 & : net_j \geq THETA \\ 0 & : sonst \end{cases} \quad (4)$$

Am Ausgang erscheint also nur genau dann der Wert '1', wenn die  $net_{nnbin}$ -Funktion einen Wert liefert, der größer oder gleich dem Wert des Schwellwertes  $THETA$  ist.

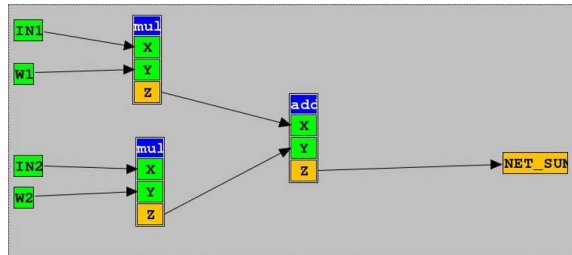


Abbildung 1: Summenfunktion 'net\_nbin' mit zwei Eingängen und Gewichten

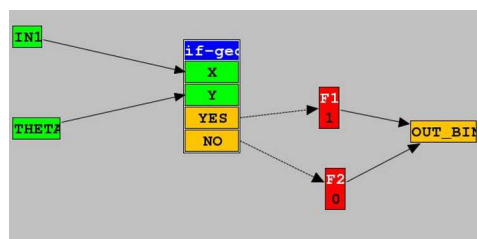


Abbildung 2: Binäre Aktivierungsfunktion

### 3 Beispiel

Bei der Modellierung des binären Neurons wird schrittweise vorgegangen, und zwar *bottom up*, d.h. zuerst werden die *Grundelemente* bereitgestellt, und dann werden aus diesen komplexere Einheiten zusammengebaut.

Im Fall des binären Neurons gibt es zwei Grundelemente: die Summenfunktion *net\_nbin* und die Aktivierungsfunktion *act\_nbin*. Beide werden so konstruiert, daß man ein neues Modell eröffnet und dieses unter dem neuen Thema *Neuron* abspeichert. Das Bild 1 zeigt das FCL-Modell einer Summenfunktion mit zwei Eingängen und zwei Gewichten.

Das Bild 2 zeigt das FCL-Modell der binären Aktivierungsfunktion. Sie hat als einen Input den Output der Summenfunktion *net\_nbin* und vergleicht diesen Werte mittels der Funktion *if\_geq* (wenn größer oder gleich). Ist der Wert der Summenfunktion größer oder gleich dem Schwellwert, dann wird eine '1' ausgegeben, sonst eine '0'.

Dann kann man das eigentliche Modell des binären Neurons bauen, indem man ein neues Modell beginnt und die vorher gebauten Modelle *net\_nbin* und *act\_nbin* als Elemente aus der Bibliothek hereinzieht. Fügt man die entsprechenden Input und Outputparameter noch hinzu und verbindet alle Ein- und Ausgänge, dann erhält man das Neuron *neuron\_bin* (vgl. das Bild 3).

Um dieses Neuron testen zu können, benötigt man jetzt noch eine geeignete

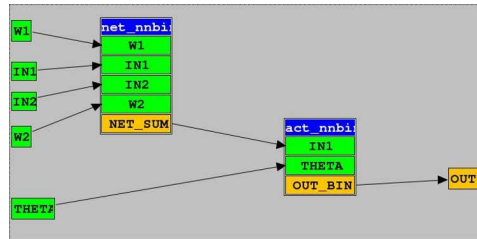


Abbildung 3: Binäres Neuron mit Summen- und Aktivierungsfunktion

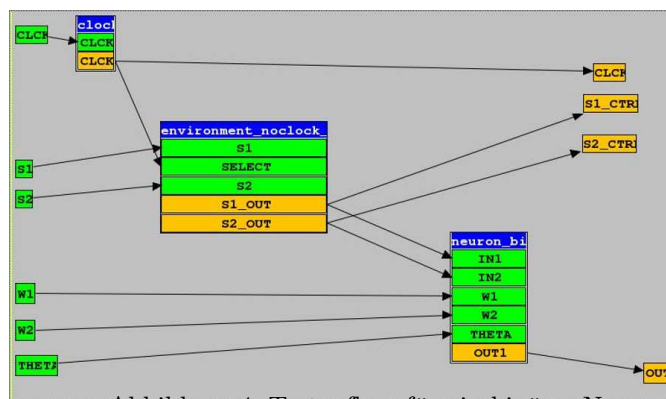


Abbildung 4: Testaufbau für ein binäres Neuron

*Testumgebung.* Im Bild 4 ist solch ein Testaufbau zu sehen. Eine Umgebung *environment\_noclock\_s2*, die über eine Uhr *clock* gesteuert wird, liefert zu jedem Zeitpunkt zwei Stimuli *S1\_OUT* und *S2\_OUT*. Diese Stimuli erscheinen an den Eingängen des binären Neurons. Zusätzlich muß man noch eingeben, welche Gewichte *W1*, *W2* man dem binären Neuron übermitteln will sowie einen Schwellwert *THETA*.

Wenn man für die Ein- und Ausgänge wie auch für die Gewichte binäre Werte  $\{0,1\}$  annimmt, dann gibt es für einen bestimmten Schwellwert *THETA* 16 verschiedene Konstellationen. In einem Testlauf kann man diese durchtesten. Für den Fall mit  $THETA = 0.5$  zeigt das Bild 5 den Output des binären Neurons für 20 Durchläufe. Man kann erkennen, daß der Schwellwert *THETA* nur dann überwunden wird, wenn wenigstens ein Eingang aktiv ist und das Gewichte den Wert '1' hatt.

Die Umgebung *environment\_noclock\_2s* (vgl. Bild 6) besteht aus zwei Listen *S1*, *S2*, jeweils realisiert als 1-dimensionale Matrizen, die für n-viele Zeitpunkt vorgegebene Werte enthalten. Jede Liste wird technisch gesehen als die 1.Spalte (mit Index '0'!) einer 1-dimensionalen Matrix, die n-viele Zeilen enthält. Während der Wert für die Spalte also immer auf '0' bleibt, wird –abhängig

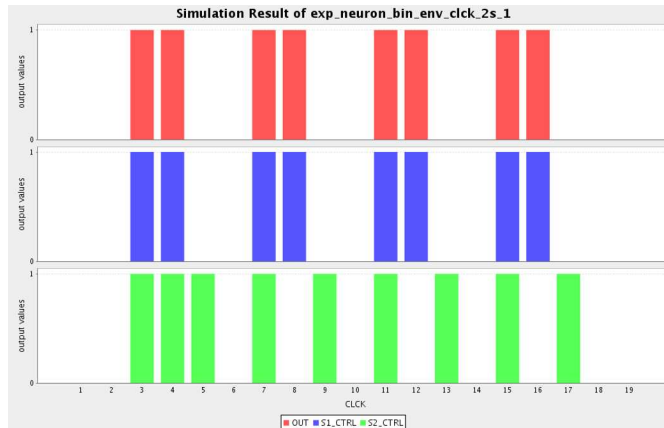


Abbildung 5: Testlauf mit einem binären Neuron

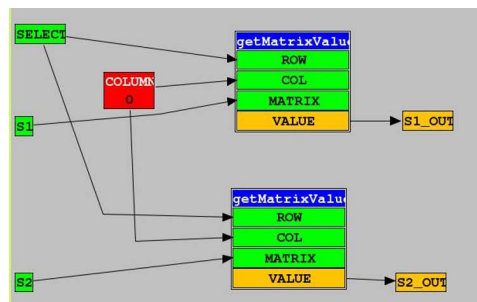


Abbildung 6: Struktur der Umgebung

vom Zeitpunkt-, eine andere Zeile gewählt. Dies geschieht über den Parameter *SELECT*, der z.B. über eine Uhr gesteuert werden kann. Abhängig von dem festen Wert  $COLUMN = 0$  und den variierenden Werten von *SELECT* können dann mit der Funktion *getMatrixValue* gezielt Werte aus den Listen gelesen werden. Man kann die Werte für die Parameter *S1*, *S2* entweder manuell in eine Matrix eingeben oder aber die Adresse für eine Datei im CSV-Format. Im letzteren Fall wird diese Datei geladen und die Werte aus dieser Datei werden in die Matrix eingesetzt.

## 4 Quellen

Siehe das Skript unter [www.uffmm.org/anns.html](http://www.uffmm.org/anns.html).