∗ Den Mauszeiger in das Eingabefeld für die Neuronenanzahl positionieren.

∗ Die Neuronenanzahl auf den gewünschten Wert korrigieren.

Enthält das Netz mehr als 7 Netzschichten, so erzeugt ProSiNN ein horizontales und ein vertikales Scrollfeld. Mit ihnen kann der Anwender das Eingabefeld für jede Netzebene erreichen.

Existieren Muster, dann kann der Benutzer die Anzahl der Neuronen in der Ein– und der Ausgabeschicht nicht verändern.

Sind im System ein zweischichtiges Netz und Muster vorhanden, dann erscheint der Button Neuronenanzahl aendern nicht in der Auswahlbox.

14.6 Der Menüpunkt „Netzwerte bearbeiten“

Mit Hilfe des Menüpunktes *Netzwerte bearbeiten* kann der Anwender auf interne Werte eines Neurons oder eines Gewichtes zugreifen. Nachdem der Menüpunkt ausgewählt wurde, erscheint z.B. das in Bild 14.8 dargestellte Popup–Fenster:

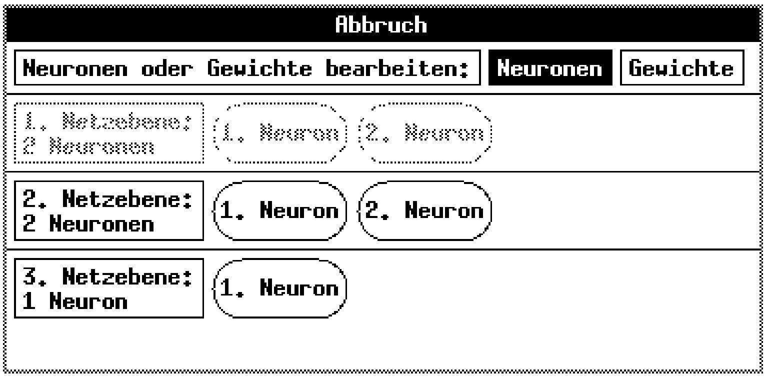


Abbildung 14.8: Das Popup–Fenster des Menüpunktes *Netz bearbeiten*

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt beendet.
- Die beiden Buttons Neuronen und Gewichte dienen als Umschalter (Toggles) zwischen den beiden Bearbeitungsmodi. Ist der Toggle Neuronen aktiv (wie in Abbildung 14.8), so kann der Benutzer interne Werte eines Neurons bearbeiten. Ist dagegen Gewichte aktiv, so sind interne Werte eines Gewichtes zugreifbar. Die Umschaltung zwischen den beiden Alternativen wird folgendermaßen durchgeführt:
  - Den Mauszeiger auf das inaktive Feld setzen.
  - Durch Drücken der linken Maustaste wird der gewünschte Modus gesetzt.
- Ist das Netz zu groß für das in Bild 14.8 dargestellte Fenster, so erzeugt ProSiNN ein horizontales und ein vertikales Scrollfeld. Mit ihnen kann der Anwender jede interne Zelle erreichen.

- Zum Berabeiten eines Neurons muß der Anwender wie folgt vorgehen:
  - Den Mauszeiger auf das gewünschte aktive Neuron setzen.
  - Durch Drücken der linken Maustaste wird ein weiteres Popup–Fenster für die entsprechende Zelle erzeugt.

  In Abbildung 14.9 wurde mit dem Mauszeiger das erste Neuron der zweiten Ebene ausgewählt:

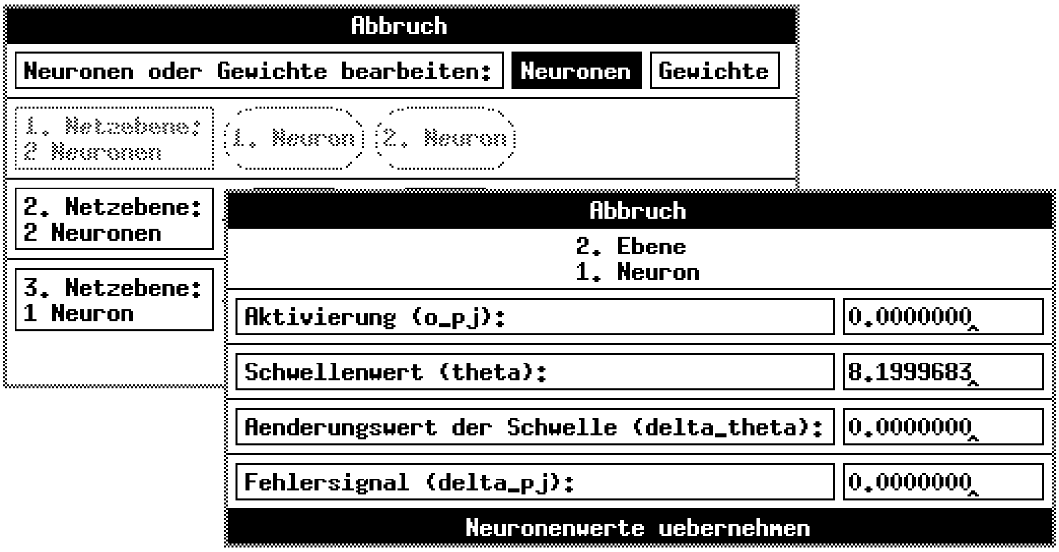


Abbildung 14.9: Das Popup–Fenster zur Bearbeitung eines Neurons

  - Existiert eine Netzebenenvariable, die diese Zelle repräsentiert, dann wird deren Bezeichner mit angegeben.
  - Durch Anklicken des Buttons $\boxed{\text{Abbruch}}$ wird dieses Popup–Fenster ohne Änderung interner Neuronenwerte beendet.
  - Interne Werte der Zelle lassen sich folgendermaßen ändern:
    - ∗ Den Mauszeiger in das entsprechende Eingabefeld positionieren.
    - ∗ Die angegebene Fließkommazahl auf den gewünschten Wert korrigieren.
  - Nachdem der Benutzer alle Änderungen vorgenommen hat, kann er die neuen Werte mit dem Button $\boxed{\text{Neuronenwerte uebernehmen}}$ gültig machen. Anschließend befindet er sich wieder in dem Fenster, das in Bild 14.8 dargestellt ist (er kann also weitere Neuronen oder Gewichte bearbeiten).

  Der Anwender kann keine Neuronen der ersten Schicht bearbeiten, da sie real nicht existieren (das System setzt diese Zellen inaktiv; siehe Abbildung 14.8)!

- Zum Bearbeiten eines Gewichtes muß der Benutzer wie folgt vorgehen:
  - Den Mauszeiger auf das gewünschte aktive Startneuron setzen und die linke Maustaste drücken. Anschließend sind alle Zellen inaktiv, die nicht in der nachfolgenden Ebene liegen.

– Nun muß der Mauszeiger auf das gewünschte Zielneuron gesetzt und die linke Maustaste erneut gedrückt werden. Nach dieser Aktion wird ein weiteres Popup–Fenster für das entsprechende Gewicht erzeugt.

In Bild 14.10 wurde das Gewicht ausgewählt, das von dem zweiten Startneuron zur ersten Zelle der zweiten Schicht verläuft:

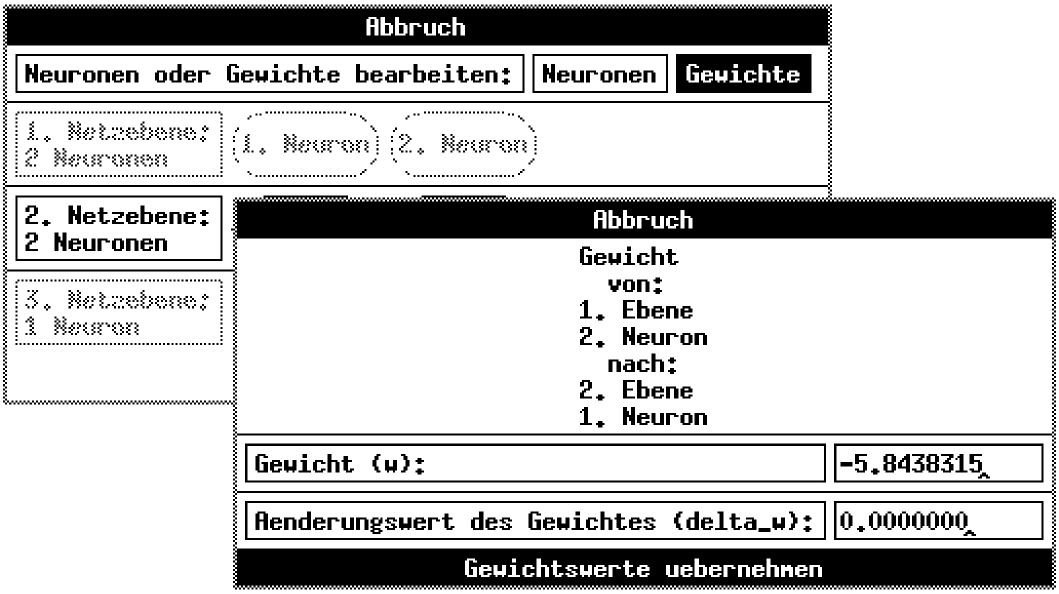


Abbildung 14.10: Das Popup–Fenster zur Bearbeitung eines Gewichtes

– Existiert eine Gewichtsvariable, die dieses Gewicht repräsentiert, dann wird deren Bezeichner mit angegeben.
– Durch Anklicken des Buttons Abbruch wird dieses Popup–Fenster ohne Änderung interner Gewichtswerte beendet.
– Interne Werte des Gewichtes lassen sich folgendermaßen ändern:
  * Den Mauszeiger in das entsprechende Eingabefeld positionieren.
  * Die angegebene Fließkommazahl auf den gewünschten Wert korrigieren.
– Nachdem der Benutzer alle Änderungen vorgenommen hat, kann er die neuen Werte mit dem Button Gewichtswerte uebernehmen gültig machen. Anschließend befindet er sich wieder in dem Fenster, das in Bild 14.8 dargestellt ist (er kann also weitere Gewichte oder Neuronen bearbeiten).

Der Anwender kann kein Gewicht bearbeiten, dessen Quelle in der der letzten Schicht liegt (sie werden vom System inaktiv gesetzt)!

14.7 Der Menüpunkt „Netz freigeben“

Will der Anwender das im System vorliegende Netz wieder löschen, so muß er den Menüpunkt *Netz freigeben* anwählen. Anschließend erscheint das in Abbildung 14.11 dargestellte Popup–Fenster:

Abbildung 14.11: Die Auswahlbox des Menüpunktes *Netz freigeben*

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt ohne Freigeben des Netzes beendet.
- Durch Auswahl des Buttons Netz freigeben wird das im System vorhandene Netz entfernt.

  Diese Vorgehensweise ist identisch mit dem Ausführen des *free_net*–Befehls in Abschnitt 9.9.

Kapitel 15

Das Menü „Muster bearbeiten“

Dieses Kapitel beschäftigt sich mit dem Menü *Muster bearbeiten*. Hier werden vier Menüpunkte angeboten, die folgendes ermöglichen:

- Trainingsmuster können geladen werden
- Testmuster lassen sich einlesen
- Muster können nachbearbeitet werden:
  - Musterinterne Werte sind veränderbar
  - Einzelne Muster können freigegeben werden
- Alle vorhandenen Muster können freigegeben werden

Die Abbildung 15.1 zeigt das Menü in der Form, die der Benutzer erhält, wenn ein Neuronales Netz im System existiert:

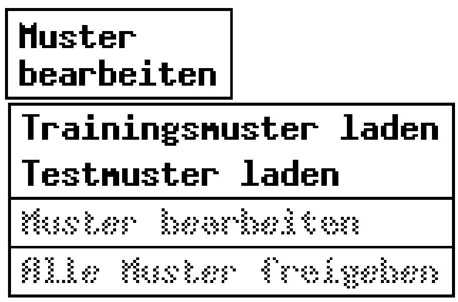


Abbildung 15.1: Das Menü *Muster bearbeiten*

Die folgenden Abschnitte führen die einzelnen Menüpunkte ein.

15.1 Der Menüpunkt „Trainingsmuster laden“

Mit dem Menüpunkt *Trainingsmuster laden* kann der Anwender Trainingsmuster einlesen.

Bem.:

- Nach der Auswahl des Menüpunktes erscheint eine Dateiauswahlbox (siehe Unterabschnitt 3.4.2), die das Laden der Trainingsmuster ermöglicht.
- Das Format der Datei, die die zu ladenden Muster enthält, muß die in Unterabschnitt 10.1.1 beschriebenen Konventionen einhalten.

15.2 Der Menüpunkt „Testmuster laden“

Um Testmuster einzulesen, muß der Benutzer den Menüpunkt *Testmuster laden* anwählen.

Bem.:

- Nach der Auswahl des Menüpunktes erscheint eine Dateiauswahlbox (siehe Unterabschnitt 3.4.2), die das Laden der Testmuster ermöglicht.
- Das Format der Datei, die die zu ladenden Muster enthält, muß die in Unterabschnitt 10.2.1 beschriebenen Konventionen einhalten.

15.3 Der Menüpunkt „Muster bearbeiten“

Mit Hilfe des Menüpunktes *Muster bearbeiten* kann der Anwender auf musterinterne Werte zugreifen oder einzelne Muster löschen. Nachdem der Menüpunkt ausgewählt wurde, erscheint z.B. das in Bild 15.2 dargestellte Popup–Fenster (hier befinden sich die Muster im System, die in den Beispielen der Unterabschnitte 10.1.1 und 10.2.1 verwendet worden sind):

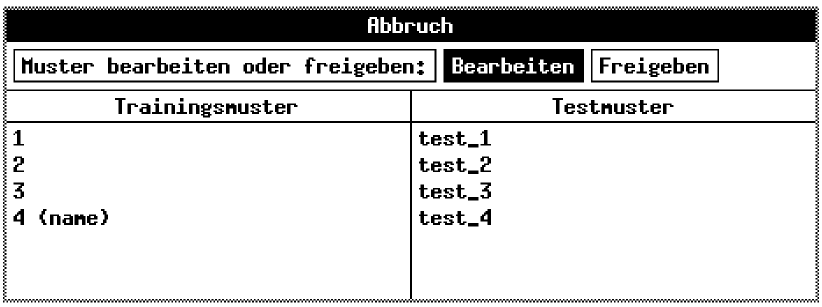


Abbildung 15.2: Das Popup–Fenster des Menüpunktes *Muster bearbeiten*

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt beendet.

- Die beiden Buttons Bearbeiten und Freigeben dienen als Umschalter (Toggles) zwischen den beiden Bearbeitungsmodi. Ist der Toggle Bearbeiten aktiv (wie in Abbildung 15.2), so kann der Benutzer interne Werte eines Musters bearbeiten. Ist dagegen Freigeben aktiv, so können einzelne Muster aus dem System entfernt werden. Die Umschaltung zwischen den beiden Alternativen wird folgendermaßen durchgeführt:
  - Den Mauszeiger auf das inaktive Feld setzen.
  - Durch Drücken der linken Maustaste wird der gewünschte Modus gesetzt.
- Passen die vorhandenen Muster nicht in das Fenster, das in Bild 15.2 dargestellt ist, so erzeugt ProSiNN ein vertikales Scrollfeld. Mit ihm kann der Anwender jedes Muster erreichen.
- Sind im System nur Trainings– oder nur Testmuster vorhanden, so besteht der gesamte Fensterinhalt aus den Mustern des entsprechenden Typs.
- Zum Berabeiten eines Musters muß der Anwender wie folgt vorgehen:
  - Den Mauszeiger auf das gewünschte Muster setzen.
  - Durch Drücken der linken Maustaste wird ein weiteres Popup–Fenster für das entsprechende Muster erzeugt.

  In Abbildung 15.3 wurde mit dem Mauszeiger das vierte Trainingsmuster mit dem Bezeichner `name` ausgewählt:

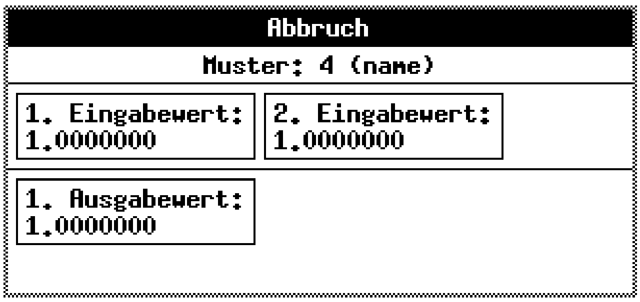


Abbildung 15.3: Das Popup–Fenster zur Bearbeitung eines Trainingsmusters

  - Durch Anklicken des Buttons Abbruch wird dieses Popup–Fenster beendet.
  - Passen nicht alle Werte des Musters in das in Bild 15.3 dargestellte Fenster, so erzeugt ProSiNN ein vertikales Scrollfeld. Mit ihm kann der Benutzer jeden Wert des Musters erreichen.
  - Interne Werte eines Musters lassen sich folgendermaßen ändern:
    - Den Mauszeiger auf den zu ändernden Wert positionieren.
    - Mit der linken Maustaste wird die entsprechende musterinterne Struktur ausgewählt und ein weiteres Popup–Fenster erzeugt.

    In Abbildung 15.4 wurde der zweite Eingabewert des vierten Trainingsmusters ausgewählt:

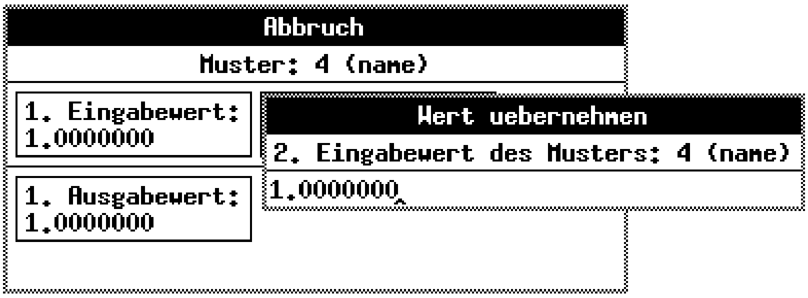


Abbildung 15.4: Das Popup–Fenster zur Bearbeitung eines musterinternen Wertes

* Zur Modifikation des Wertes muß der Mauszeiger auf das Eingabefeld gesetzt und der angegebene Fließkommawert korrigiert werden.
* Durch Anklicken des Buttons $\boxed{\text{Wert uebernehmen}}$ wird das aktuell vorliegende Datum übernommen und das Popup–Fenster wieder gelöscht.

- Zum Freigeben eines Gewichtes muß der Benutzer wie folgt vorgehen:
  - Den Mauszeiger auf das gewünschte Muster setzen.
  - Durch Drücken der linken Maustaste wird das angegebene Muster aus der vorliegenden Liste und dem System entfernt.
  - Wird ein Trainingsmuster freigegeben, dann sind die Zugriffsschlüssel der nachfolgenden Muster anschließend um eins kleiner.
  - Ein Bezeichner, der das freizugebende Muster repräsentiert, ist im Anschluß nicht mehr definiert.

  In Bild 15.5 ist die Situation dargestellt, die nach dem Freigeben des Testmusters `test_2` vorliegt:

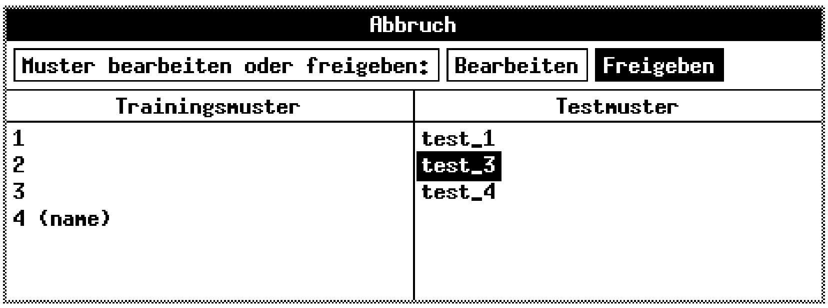


Abbildung 15.5: Das Popup–Fenster nach dem Entfernen eines Testmusters

15.4 Der Menüpunkt „Alle Muster freigeben“

Sollen alle im System vorhandenen Muster entfernt werden, dann muß der Menüpunkt *Alle Muster freigeben* angewählt werden. Anschließend erscheint das in Abbildung 15.6 dargestellte Popup–Fenster:

Abbildung 15.6: Die Auswahlbox des Menüpunktes *Alle Muster freigeben*

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt ohne Freigeben aller Muster beendet.
- Durch Auswahl des Buttons Alle Muster freigeben werden alle im System vorhandenen Muster eliminiert.

  Diese Vorgehensweise ist identisch mit dem Ausführen des *free_all_pattern*–Befehls in Unterabschnitt 10.5.2.

Kapitel 16

Das Menü „Netz trainieren“

Dieses Kapitel beschäftigt sich mit dem Menü *Netz trainieren*, das in Abbildung 16.1 dargestellt ist (diese Form erhält man, wenn im System ein Neuronales Netz und mindestens ein Trainingsmuster existieren):

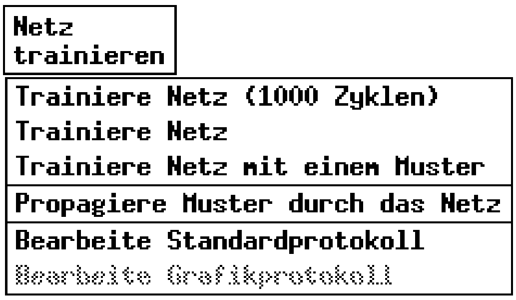


Abbildung 16.1: Das Menü *Netz trainieren*

Hier werden sechs Menüpunkte angeboten, die folgendes ermöglichen:

- Das Netz kann bis zu einer vorgegebenen maximalen Anzahl von Trainingszyklen trainiert werden
- Der Benutzer kann das Netz bis zu einer von ihm angegebenen maximalen Anzahl Zyklen trainieren lassen
- Das Netz kann mit einem beliebigen Muster trainiert werden
- Die Netzantwort eines Musters kann berechnet und ausgegeben werden:
  - Bei einem Trainingsmuster werden die Ein– und die Ausgabewerte (und die entsprechenden Ausgabefehler) angegeben
  - Bei einem Testmuster werden die Ein– und die Ausgabewerte angezeigt
- Textprotokolle können bearbeitet werden:
  - Ein Textprotokoll kann neu erzeugt werden

  - Alle Textprotokolle können entfernt werden

- Grafikprotokolle können bearbeitet werden:
  - Ein Grafikprotokoll kann neu generiert werden
  - Alle Grafikrotokolle können eliminiert werden

Die folgenden Abschnitte führen die einzelnen Menüpunkte ein.

16.1 Der Menüpunkt „Trainiere Netz (1000 Zyklen)“

Will der Anwender das Training maximal bis zu einer vorgegebenen Anzahl von Zyklen ausführen lassen, so muß er den Menüpunkt *Trainiere Netz (1000 Zyklen)* anwählen.

Bem.:

- Nach der Auswahl dieses Punktes wird das Netz maximal bis zu der angegebenen Anzahl Zyklen trainiert.
- Es wird das Lernverfahren verwendet, das in Abschnitt 2.3 eingeführt wurde.
- Das Netz wird solange trainiert, bis eine der folgenden Situationen eintritt:
  - Das Fehlermaß **act_error** (das auf die Art und Weise berechnet wird, die durch die Variable **error_mode** festgelegt ist) unterschreitet die vorgegebene Fehlermaßgrenze **error_limit**.

    Wenn diese Abbruchbedingung eintritt, ist das Lernziel erreicht worden!
  - Die Anzahl der bisher ausgeführten Trainingszyklen **act_iterations** überschreitet die maximale Anzahl der auszuführenden Zyklen.

    Tritt diese Abbruchbedingung ein, so ist das Lernziel nicht erreicht worden!
  - Der Benutzer bricht die Ausführung mit der Tastenkombination `Control-C` im Ein/Ausgabefenster ab.
- Parallel zum Training kann das System eine beliebige Anzahl von Protokollen in das Ein/Ausgabefenster (siehe Abschnitt 16.5) und in Grafiken (siehe Abschnitt 16.6) ausgeben. Die Protokolldefinitionen müssen vor Aufruf dieses Menüpunktes definiert werden.
- Will der Anwender beim Netztraining einen anderen Fehlermaßmodus benutzen, dann empfiehlt sich folgende Strategie:

  Das Netz wird solange mit dem *quadratischen* Fehlermaßmodus trainiert, bis die vorgegebene Fehlermaßgrenze **error_limit** unterschritten ist. Im Anschluß wird dann der gewünschte Fehlermaßmodus gesetzt und das Netz erneut bis zur Lösung des Problems trainiert.

  Durch diese Vorgehensweise erzielt man i.allg. eine schnellere Konvergenz des Lernverfahrens!

- Hat der Anwender im Menüpunkt *Trainiere Netz* oder im Menüpunkt *Trainiere Netz mit einem Muster* einen anderen Wert als 1000 für die maximale Anzahl der Zyklen eingegeben (siehe Abschnitt 16.2 und 16.3), so erscheint hier in diesem Menüpunkt als Text der dort angegebene Wert (anstatt 1000). Wurde dort z.B. bis zu einer maximalen Anzahl von 500 Zyklen trainiert, so heißt dieser Menüpunkt anschließend *Trainiere Netz (500 Zyklen).*

16.2 Der Menüpunkt „Trainiere Netz“

Wenn der Benutzer das Netz maximal bis zu einer von ihm eingegebenen Anzahl von Zyklen trainieren lassen will, so muß er den Menüpunkt *Trainiere Netz* anwählen. Danach erscheint das Popup–Fenster, das in Bild 16.2 dargestellt ist:

Abbildung 16.2: Das Popup–Fenster des Menüpunktes *Trainiere Netz*

Bem.:

- Durch Anklicken des Buttons $\boxed{\text{Abbruch}}$ wird der Menüpunkt ohne Training des Netzes beendet.
- Die maximale Anzahl der auszuführenden Trainingszyklen kann der Anwender folgendermaßen ändern:
  - Den Mauszeiger in das Eingabefeld für die maximale Anzahl der Zyklen positionieren.
  - Die dort angegebene ganze Zahl auf den gewünschten Wert korrigieren.
- Durch Anklicken des Buttons $\boxed{\text{Trainiere Netz}}$ wird das Netz maximal bis zu der angegebenen Anzahl Zyklen trainiert.
- Es wird das Lernverfahren verwendet, das in Abschnitt 2.3 eingeführt wurde.
- Das Netz wird solange trainiert, bis eine der folgenden Situationen eintritt:
  - Das Fehlermaß **act_error** (das auf die Art und Weise berechnet wird, die durch die Variable **error_mode** festgelegt ist) unterschreitet die vorgegebene Fehlermaßgrenze **error_limit**.

    Wenn diese Abbruchbedingung eintritt, ist das Lernziel erreicht worden!
  - Die Anzahl der bisher ausgeführten Trainingszyklen **act_iterations** überschreitet die maximale Anzahl der auszuführenden Zyklen.

    Tritt diese Abbruchbedingung ein, so ist das Lernziel nicht erreicht worden!

  - Der Benutzer bricht die Ausführung mit der Tastenkombination `Control-C` im Ein/Ausgabefenster ab.

- Parallel zum Training kann das System eine beliebige Anzahl von Protokollen in das Ein/Ausgabefenster (siehe Abschnitt 16.5) und in Grafiken (siehe Abschnitt 16.6) ausgeben. Die Protokolldefinitionen müssen vor Aufruf dieses Menüpunktes definiert werden.

- Will der Anwender beim Netztraining einen anderen Fehlermaßmodus benutzen, dann empfiehlt sich folgende Strategie:

  Das Netz wird solange mit dem *quadratischen* Fehlermaßmodus trainiert, bis die vorgegebene Fehlermaßgrenze **error_limit** unterschritten ist. Im Anschluß wird dann der gewünschte Fehlermaßmodus gesetzt und das Netz erneut bis zur Lösung des Problems trainiert.

  Durch diese Vorgehensweise erzielt man i.allg. eine schnellere Konvergenz des Lernverfahrens!

16.3 Der Menüpunkt „Trainiere Netz mit einem Muster"

Will der Anwender das Netz mit einem einzigen Muster und maximal bis zu einer von ihm angegebenen Anzahl von Zyklen trainieren lassen, so muß er den Menüpunkt *Trainiere Netz mit einem Muster* anwählen. Anschließend erscheint z.B. das in Abbildung 16.3 dargestellte Popup–Fenster (hier befinden sich die Trainingsmuster im System, die im Beispiel des Unterabschnitts 10.1.1 verwendet worden sind):

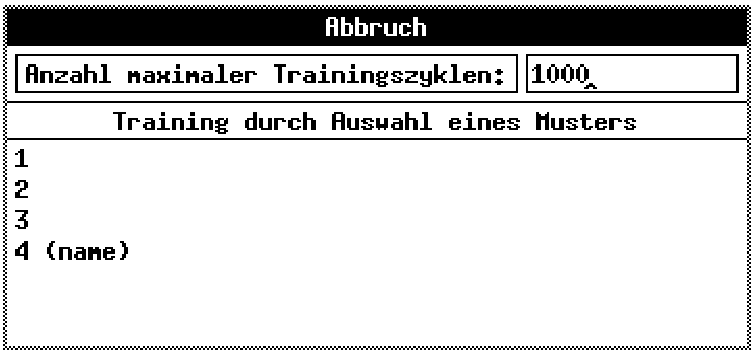


Abbildung 16.3: Das Popup–Fenster des Menüpunktes *Trainiere Netz mit einem Muster*

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt beendet.

- Die maximale Anzahl der auszuführenden Trainingszyklen kann der Anwender folgendermaßen ändern:

  - Den Mauszeiger in das Eingabefeld für die maximale Anzahl der Zyklen positionieren.

  - Die dort angegebene ganze Zahl auf den gewünschten Wert korrigieren.

- Um das Training mit einem bestimmten Muster zu starten, muß der Benutzer wie folgt vorgehen:

  - Den Mauszeiger auf das gewünschte Trainingsmuster setzen.
  - Durch Drücken der linken Maustaste wird das Training gestartet.
  - Nach der Beendigung des Trainings bleibt das Popup–Fenster aktiv. Es können also weitere Muster trainiert werden.

  Passen die existierenden Trainingsmuster nicht in das in Bild 16.3 dargestellte Fenster, dann erzeugt ProSiNN ein vertikales und ein horizontales Scrollfeld. Mit ihnen können alle Trainingsmuster erreicht werden.

- Es wird das Lernverfahren verwendet, das in Abschnitt 2.3 eingeführt wurde.

- Das Netz wird solange mit dem angegebenen Muster trainiert, bis eine der folgenden Situationen eintritt:

  - Das Fehlermaß **act_error** (das auf die Art und Weise berechnet wird, die durch die Variable **error_mode** festgelegt ist) unterschreitet die vorgegebene Fehlermaßgrenze **error_limit**.

    Wenn diese Abbruchbedingung eintritt, ist das Lernziel erreicht worden!
  - Die Anzahl der bisher ausgeführten Trainingszyklen **act_iterations** überschreitet die maximale Anzahl der auszuführenden Zyklen.

    Tritt diese Abbruchbedingung ein, so ist das Lernziel nicht erreicht worden!
  - Der Benutzer bricht die Ausführung mit der Tastenkombination `Control-C` im Ein/Ausgabefenster ab.

- Parallel zum Training kann das System eine beliebige Anzahl von Protokollen in das Ein/Ausgabefenster (siehe Abschnitt 16.5) und in Grafiken (siehe Abschnitt 16.6) ausgeben. Die Protokolldefinitionen müssen vor Aufruf dieses Menüpunktes definiert werden.

- Mit Hilfe dieses Menüpunktes läßt sich das Netz an ein bestimmtes Muster anpassen. Dies ist evtl. wünschenswert, wenn das Netz Schwierigkeiten beim Lernen *eines* Musters hat.

16.4 Der Menüpunkt „Propagiere Muster durch das Netz“

Um die Netzantwort eines bestimmten Musters anzuzeigen, muß der Menüpunkt *Propagiere Muster durch das Netz* angewählt werden. Danach erscheint z.B. das in Abbildung 16.4 dargestellte Popup–Fenster (hier befinden sich die Trainings– und Testmuster im System, die in den Beispielen der Unterabschnitte 10.1.1 und 10.2.1 verwendet worden sind; das vorliegende Netz stellt eine Lösung des `AND`–Problems dar):

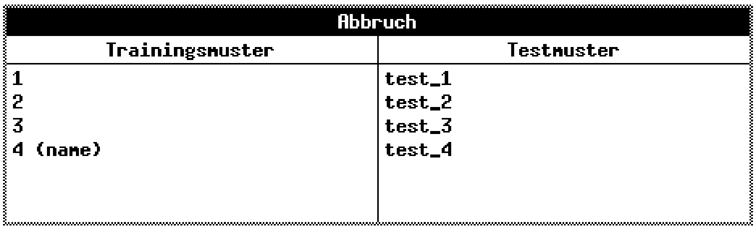


Abbildung 16.4: Das Popup–Fenster des Menüpunktes *Propagiere Muster durch das Netz*

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt beendet.
- Passen die existierenden Muster nicht in das Fenster, das in Bild 16.4 dargestellt ist, so erzeugt ProSiNN ein vertikales Scrollfeld. Mit ihm kann der Anwender jedes Muster erreichen.
- Sind im System nur Trainings– oder nur Testmuster vorhanden, so besteht der gesamte Fensterinhalt aus den Mustern des entsprechenden Typs.
- Um die Netzantwort eines Musters anzuzeigen, muß der Benutzer wie folgt vorgehen:
  - Den Mauszeiger auf das gewünschte Muster setzen.
  - Durch Drücken der linken Maustaste wird ein weiteres Popup–Fenster mit dem entsprechenden Muster und der Netzantwort erzeugt.
- In Abbildung 16.5 wurde mit dem Mauszeiger das vierte Trainingsmuster mit dem Bezeichner `name` ausgewählt. Das Bild 16.6 ist durch die Auswahl des Testmusters mit dem Namen `test_1` entstanden.

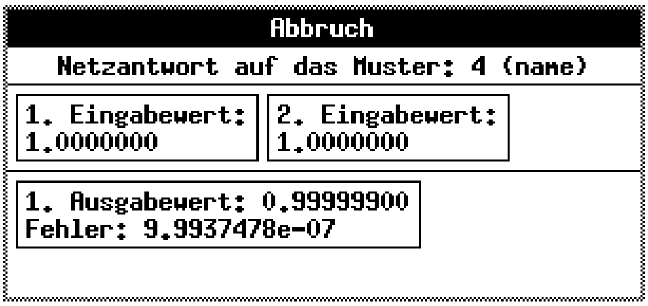


Abbildung 16.5: Das Popup–Fenster zur Anzeige der Netzantwort eines Trainingsmusters

  - Durch Anklicken des Buttons Abbruch gelangt man wieder zu dem Fenster, das eine weitere Musterauswahl ermöglicht.
  - Passen die Ein– oder Ausgabewerte der Muster nicht in das in Abbildung 16.5 (oder in Bild 16.6) dargestellte Fenster, so erzeugt ProSiNN ein horizontales Scrollfeld. Mit ihm kann der Anwender jede Komponente des Musters oder der Netzantwort erreichen.
  - Bei einem Trainingsmuster wird neben der Netzantwort auch noch der Fehler jedes Ausgabeneurons mit dem vorgegebenen Zielwert angegeben.

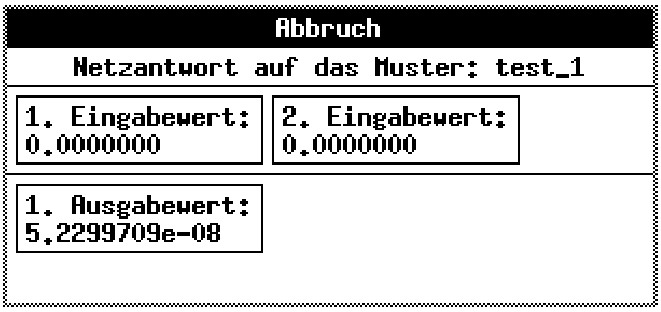


Abbildung 16.6: Das Popup–Fenster zur Anzeige der Netzantwort eines Testmusters

16.5 Der Menüpunkt „Bearbeite Standardprotokoll“

Will der Benutzer ein Textprotokoll in das Ein/Ausgabefenster definieren oder alle Textprotokolle löschen, so muß er den Menüpunkt *Bearbeite Standardprotokoll* anwählen. Existiert noch kein Textprotokoll, dann erscheint das in Abbildung 16.7 dargestellte Popup–Fenster:

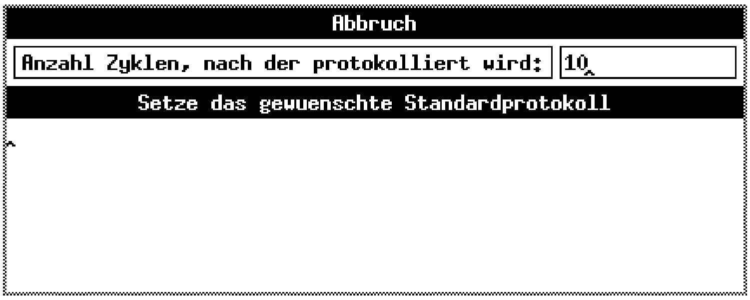


Abbildung 16.7: Das Popup–Fenster des Menüpunktes *Bearbeite Standardprotokoll*

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt ohne Generierung eines Textprotokolls beendet.

- Die Anzahl Zyklen, nach der eine Protokollzeile ausgegeben werden soll, läßt sich wie folgt ändern:

  – Den Mauszeiger auf das Eingabefeld für den Zyklenabstand setzen.

  – Die dort angegebene ganze Zahl auf den gewünschten Wert korrigieren.

  Sei $\mathbf{x}$ die Zahl, die den Zyklenabstand beschreibt. Dann wird beim Training eine Protokollzeile nach den folgenden Lernzyklen ausgegeben: $n * \mathbf{x}$, wobei $n \in \{1, 2, 3, \ldots\}$.

- Mit dem Button Setze das gewuenschte Standardprotokoll wird der darunterliegende Text als Protokoll definiert:

  – Der Protokolltext muß folgende Syntax einhalten:

protocol_text ::= **protocol_parameter**
{ “,“ **protocol_parameter** }.
protocol_parameter ::= **int_val** | **double_val** | **array_val** | **funct_val** |
string_val.

protocol_parameter definiert ein Element der Protokollzeile. Die Anzahl dieser Argumente ist beliebig (sofern sie größer als 1 ist). Die Parameter werden bei der Protokollierung in der Reihenfolge ihrer Definition ausgegeben:

* **int_val** muß eine Konstante oder Variable vom Typ *int* sein.
* **double_val** ist eine Konstante oder Variable vom Typ *double*.
* **array_val** muß folgende Konvention einhalten:
  array_val ::= **array_name**“[“ **int_val** “]“.
  array_name ::= **„Bezeichner einer Feldvariablen“.**
    · **array_name** muß der Bezeichner einer Feldvariablen vom Typ *int* oder *double* sein.
    · **int_val** ist eine Konstante oder Variable vom Typ *int*. Dieser Wert muß bei der Auswertung des Protokolltextes und während des Protokollierens ein gültiges Feldelement repräsentieren.
* **funct_val** muß folgender Syntax genügen:
  funct_val ::= **funct_name**“(“ **funct_arguments** “)“.
  funct_name ::= **„Bezeichner einer Funktionsvariablen“.**
  funct_arguments ::= **val** { “,“ **val** }.
  val ::= **int_val** | **double_val.**
    · **funct_name** ist der Bezeichner einer Funktionsvariablen vom Typ *int* oder *double*.
    · **funct_arguments** setzt die Funktionsargumente. Für jedes Funktionsargument muß **val** stehen; dabei darf **val** eine Konstante oder Variable vom Typ *int* oder *double* sein.
* **string_val** muß eine Konstante oder Variable vom Typ *string* sein.

– ProSiNN führt am Ende einer Protokollzeile keinen Zeilenumbruch durch. Aus diesem Grund sollte der letzte Parameter immer **EOL** sein.

– Wenn ein zu protokollierendes Element während des Trainings nicht mehr ansprechbar ist, bricht ProSiNN das Lernverfahren mit einer Fehlermeldung ab.

– Der Benutzer kann beliebig viele Textprotokolle generieren. Man beachte aber, jedes weitere Protokoll verlangsamt die Durchführung des Trainings!

– Paßt der eingegebene Text nicht mehr in das in Bild 16.7 dargestellte Texteingabefenster, so erzeugt ProSiNN ein vertikales Scrollfeld. Mit ihm kann der Anwender jeden Teil des Protokolltextes betrachten und ändern.

– Entdeckt das System einen Fehler im Protokolleingabetext, so erscheint der invertierte Fehlerausgabebutton (mit der Fehlermeldung), der über dem Ein/Ausgabefenster plaziert wird (siehe Unterabschnitt 3.4.2). Nachdem der Benutzer den Button mit einer Maustaste angeklickt hat, ist die Fehlerstelle im Eingabefenster des Protokolltextes invertiert dargestellt.

  In Abbildung 16.8 ist statt der vordefinierten Konstanten **EOL** der dem System unbekannte Bezeichner **eol** eingegeben worden!

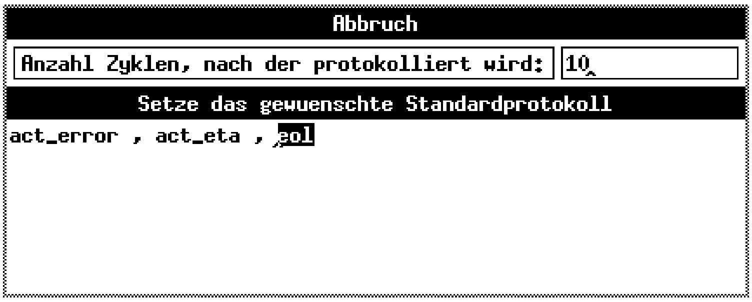


Abbildung 16.8: Die Fehlerdarstellung beim Definieren eines Textprotokolls

- Wählt der Anwender den Menüpunkt *Bearbeite Standardprotokoll* aus und es existiert ein Textprotokoll im System, dann erscheint das in Abbildung 16.9 dargestellte Auswahlfenster:

Abbildung 16.9: Das Auswahlfenster des Menüpunktes *Bearbeite Standardprotokoll*

  - Durch Anklicken des Buttons Abbruch wird der Menüpunkt beendet.
  - Wenn der Benutzer alle existierenden Textprotokolle löschen will, so muß er den Button Loesche alle Textprotokolle anwählen.
  - Mit dem Button Erzeuge ein neues Standardprotokoll erhält der Anwender das in Bild 16.7 dargestellte Popup–Fenster. Mit dessen Hilfe kann ein neues Protokoll generiert werden (siehe oben).

- Soll ein Textprotokoll in eine Datei geschrieben werden, so muß auf das in Unterabschnitt 12.1.1 eingeführte *protocol*–Kommando zurückgegriffen werden.

16.6 Der Menüpunkt „Bearbeite Grafikprotokoll“

Wenn im System offene Grafikfenster existieren und der Benutzer ein Grafikprotokoll erzeugen oder alle Grafikprotokolle löschen will, so muß er den Menüpunkt *Bearbeite Grafikprotokoll* anwählen. War noch kein Grafikprotokoll vorhanden, dann erscheint das in Abbildung 16.10 dargestellte Popup–Fenster.

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt ohne Generierung eines Grafikprotokolls beendet.

- Die Grafik, in die protokolliert werden soll, läßt sich folgendermaßen ändern:
  - Den Mauszeiger auf das Eingabefeld für den Grafikkontext setzen.

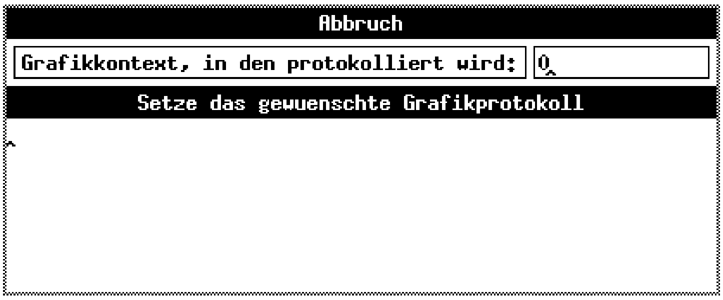


Abbildung 16.10: Das Popup–Fenster des Menüpunktes *Bearbeite Grafikprotokoll*

  - Die dort angegebene ganze Zahl auf den gewünschten Grafikkontext korrigieren.

- Mit dem Button $\boxed{\text{Setze das gewuenschte Grafikprotokoll}}$ wird der darunterliegende Text als Protokoll definiert:

  - Der Protokolltext muß sich an folgende Syntax halten:

    protocol_text ::= **protocol_parameter**
    { “,“ **protocol_parameter** }.
    protocol_parameter ::= [**dotted_specification**]
    (**int_val** | **double_val** | **array_val** | **funct_val**).

    protocol_parameter definiert ein Element, das in das angegebene Grafikfenster ausgegeben werden soll. Die Anzahl der **protocol_parameter** ist beliebig (sofern sie größer als 1 ist):

    * **dotted_specification** beschreibt die optionale Angabe einer Strichlierung (siehe Abschnitt 8.4), mit der dieser Parameter ausgegeben werden soll.
    * **int_val** muß eine Konstante oder Variable vom Typ *int* sein.
    * **double_val** ist eine Konstante oder Variable vom Typ *double*.
    * **array_val** muß folgende Konvention einhalten:

      array_val ::= **array_name**“*[*“ **int_val** “*]*“.
      array_name ::= **„Bezeichner einer Feldvariablen“.**

      · **array_name** muß der Bezeichner einer Feldvariablen vom Typ *int* oder *double* sein.

      · **int_val** ist eine Konstante oder Variable vom Typ *int*. Dieser Wert muß bei der Auswertung des Protokolltextes und während des Protokollierens ein gültiges Feldelement repräsentieren.

    * **funct_val** muß folgender Syntax genügen:

      funct_val ::= **funct_name**“*(*“ **funct_arguments** “*)*“.
      funct_name ::= **„Bezeichner einer Funktionsvariablen“.**
      funct_arguments ::= **val** { “,“ **val** }.
      val ::= **int_val** | **double_val.**

      · **funct_name** ist der Bezeichner einer Funktionsvariablen vom Typ *int* oder *double*.

    · **funct_arguments** setzt die Funktionsargumente. Für jedes Funktionsargument muß **val** stehen; dabei darf **val** eine Konstante oder Variable vom Typ *int* oder *double* sein.

  – Die Anzahl der Lernzyklen wird beim Protokollieren durch die Abszissenachse repräsentiert.
  – Durch die Breite des Grafikfensters und des Weltkoordinatensystems (WKS) errechnet ProSiNN die Zyklenanzahl, nach der ein Element ausgegeben werden muß.
  – Wird beim Ausgeben der rechte Rand des Weltkoordinatensystems erreicht, dann löscht ProSiNN den Fensterinhalt, zeichnet neue Koordinatenachsen ein (dabei wird der rechte Rand des WKS zum neuen linken Rand des WKS; die Breite des WKS bleibt erhalten; der untere Wert der Ordinate des WKS wird 0.0 und der obere Rand wird durch das letzte aktuelle Fehlermaß bestimmt) und fährt mit dem Lernen und Protokollieren fort.
  – Wenn ein zu protokollierendes Element während des Trainings nicht mehr ansprechbar ist, bricht ProSiNN das Lernverfahren mit einer Fehlermeldung ab.
  – Der Benutzer kann beliebig viele Grafikprotokolle erzeugen. Man beachte aber, jedes weitere Protokoll verlangsamt die Durchführung des Trainings!
  – Paßt der eingegebene Text nicht mehr in das in Bild 16.10 dargestellte Texteingabefenster, so erzeugt ProSiNN ein vertikales Scrollfeld. Mit ihm kann der Anwender jeden Teil des Protokolltextes betrachten und ändern.
  – Entdeckt das System einen Fehler im Eingabefeld des Textfensters, so wird die Fehlerstelle invertiert dargestellt (siehe die entsprechende Bemerkung im letzten Abschnitt und die Abbildung 16.8).

- Wählt der Anwender den Menüpunkt *Bearbeite Grafikprotokoll* aus und es existiert ein Grafikprotokoll im System, dann erscheint das in Abbildung 16.11 dargestellte Auswahlfenster:

Abbildung 16.11: Das Auswahlfenster des Menüpunktes *Bearbeite Grafikprotokoll*

  – Durch Anklicken des Buttons Abbruch wird der Menüpunkt beendet.
  – Wenn der Benutzer alle existierenden Grafikprotokolle löschen will, so muß er den Button Loesche alle Grafikprotokolle anwählen.
  – Mit dem Button Erzeuge ein neues Grafikprotokoll erhält der Anwender das in Abbildung 16.10 dargestellte Popup–Fenster. Mit dessen Hilfe kann ein neues Protokoll generiert werden (siehe oben).

Kapitel 17

Das Menü „Systemwerte bearbeiten“

Dieses Kapitel beschäftigt sich mit dem Menü *Systemwerte bearbeiten*, das in Abbildung 17.1 dargestellt ist (diese Form erhält man, wenn die vordefinierte sigmoide Aktivierungsfunktion verwendet wird):

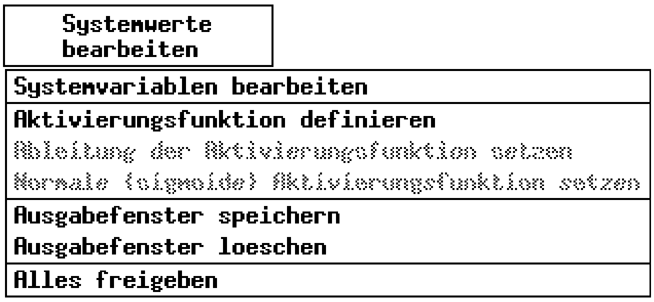


Abbildung 17.1: Das Menü *Systemwerte bearbeiten*

Hier werden sieben Menüpunkte angeboten, die folgendes ermöglichen:

- Die Systemvariablen können verändert werden:
  - Deren Werte sind vom Benutzer modifizierbar
  - Den Variablen können die Defaultwerte zugewiesen werden
- Die Aktivierungsfunktion kann neu definiert und später auch interaktiv nachbearbeitet werden
- Die Ableitung der Aktivierungsfunktion kann verändert werden
- Die normale (sigmoide) Aktivierungsfunktion läßt sich wieder setzen
- Der Inhalt des Ein/Ausgabefensters kann in einer Datei gespeichert (und anschließend evtl. gelöscht) werden

- Der Inhalt des Ein/Ausgabefensters kann gelöscht werden
- Das ganze System kann in den Zustand versetzt werden, der beim Start von ProSiNN vorlag

Die folgenden Abschnitte führen die einzelnen Menüpunkte ein.

17.1 Der Menüpunkt „Systemvariablen bearbeiten“

Der Anwender kann den Menüpunkt *Systemvariablen bearbeiten* anwählen, wenn er den Inhalt einiger ausgewählter Variablen betrachten oder ändern will. Anschließend erscheint das Popup–Fenster, das in Bild 17.2 dargestellt ist[1].

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt beendet.
- Will der Benutzer die Defaultwerte an die Systemvariablen zuweisen, so muß der Button Initialisiere mit Defaultwerten ausgewählt werden. Dann werden die folgenden vordefinierten Variablen auf die angegebenen Werte gesetzt:
  - **sum_iterations** = **act_iterations** = 0
  - **error_mode** = **quadratisch**
  - **error_limit** = 0.001
  - **act_error** = **old_error** = 0.0
  - **eta** = **act_eta** = 0.2
  - **alpha** = **act_alpha** = 0.9
  - **fail_limit** = 1.04
  - **phi** = 1.05
  - **beta** = 0.7
  - **floatpattern** = **on**
  - **pat_min** = 0.0 und **pat_max** = 1.0
  - **precision** = 8

  Die nachfolgenden (nicht in Abbildung 17.2 dargestellten) Variablen werden außerdem noch zurückgesetzt:
  - **theta_min** = −0.5 und **theta_max** = 0.5
  - **w_min** = −0.5 und **w_max** = 0.5
- Mit der Hilfe des Buttons Setze die angegebenen Werte kann der Anwender von ihm eingegebene Werte an die Variablen zuweisen.

[1] Bei dieser Abbildung handelt es sich um eine Bildmontage. Normalerweise stellt das erzeugte Fenster nur einen Teil des hier sichtbaren Bereichs dar

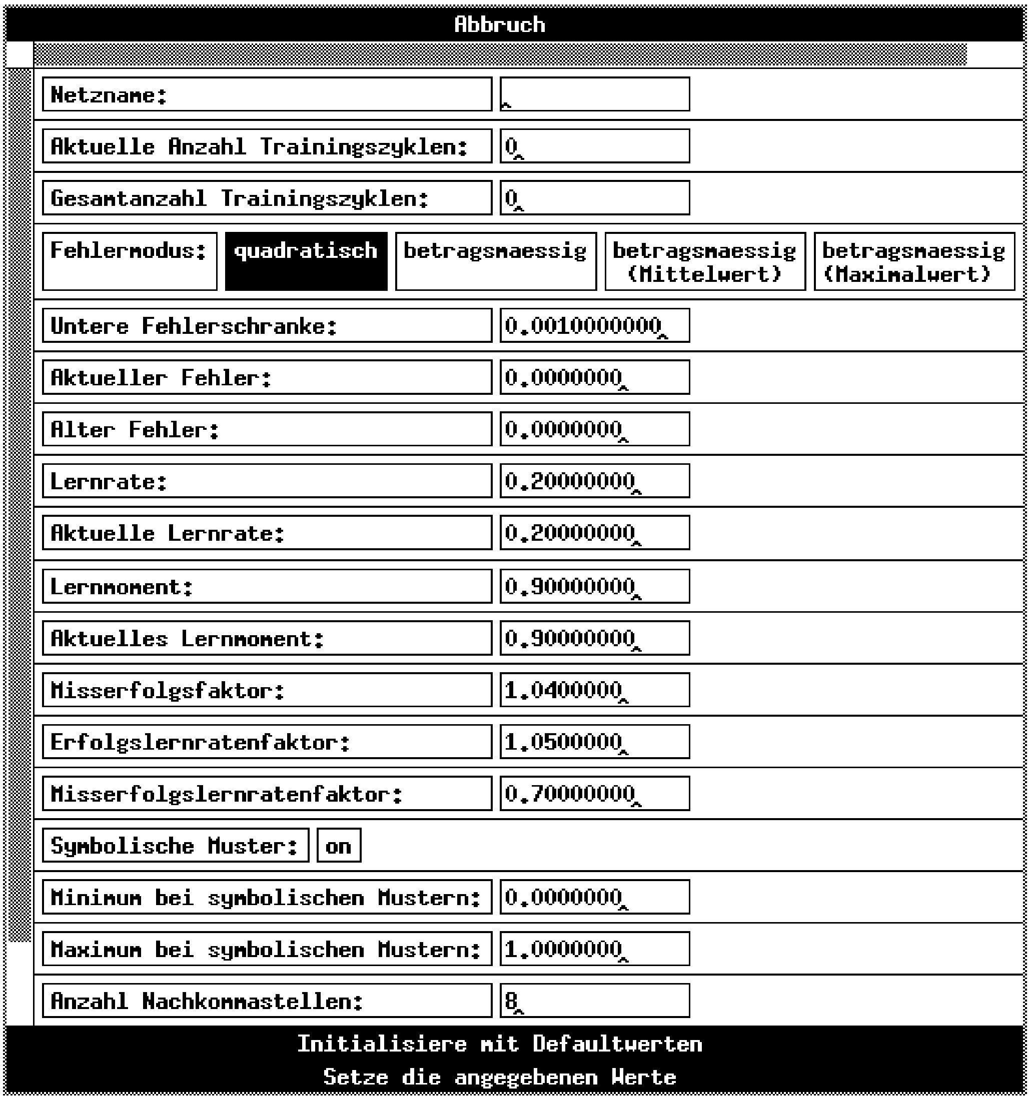


Abbildung 17.2: Das Popup–Fenster des Menüpunktes *Systemwerte bearbeiten*

- Die im Bild dargestellten Systemvariablen lassen sich folgendermaßen ändern:
  * Den Mauszeiger in das Eingabefeld des zu modifizierenden Wertes setzen.
  * Den dort angegebenen Wert korrigieren.
- Die Buttons $\boxed{\text{quadratisch}}$, $\boxed{\text{betragsmaessig}}$, $\boxed{\text{betragsmaessig (Mittelwert)}}$ und $\boxed{\text{betragsmaessig (Maximalwert)}}$ dienen als Umschalter (Toggles) zwischen den verschiedenen Fehlermaßmodi. Will der Benutzer den Fehlermaßmodus ändern, so muß er wie folgt vorgehen:
  * Den Mauszeiger auf den gewünschten inaktiven Fehlermodus setzen.
  * Durch Drücken der linken Maustaste wird der Modus gesetzt.
  * Anschließend hat die Variable **error_mode** einen der folgenden Werte:

· **error_mode** = **quadratisch**, wenn quadratisch aktiv ist.

· **error_mode** = **betragsmaessig**, wenn betragsmaessig aktiv ist.

· **error_mode** = **mittel_betrag**, wenn betragsmaessig (Mittelwert) aktiv ist.

· **error_mode** = **max_betrag**, wenn betragsmaessig (Maximalwert) aktiv ist.

– Der Button on (in Bild 17.2 befindet er sich rechts des Textes *Symbolische Muster:*) dient als Umschalter, der den Modus beim Einlesen von Trainings– oder Testmustern setzt. Zu dessen Änderung geht man folgendermaßen vor:

∗ Den Mauszeiger auf den Button setzen.

∗ Durch Drücken der linken Maustaste wird der Modus gewechselt.

∗ Anschließend hat die Variable **floatpattern** einen der folgenden Werte:

· **floatpattern** = **on**, wenn on inaktiv ist.

· **floatpattern** = **off**, wenn on aktiv ist.

17.2 Der Menüpunkt „Aktivierungsfunktion definieren“

Will der Anwender eine andere Aktivierungsfunktion benutzen oder die neu definierte Aktivierungsfunktion ändern, so kann er den Menüpunkt *Aktivierungsfunktion definieren* anwählen. Anschließend erscheint z.B. das in Abbildung 17.3 dargestellte Popup–Fenster:

Abbildung 17.3: Das Popup–Fenster des Menüpunktes *Aktivierungsfunktion definieren*

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt ohne Änderung der vorliegenden Aktivierungsfunktion beendet.

- Mit dem Button Eingegebene Aktivierungsfunktion setzen wird der darüberliegende Text als Aktivierungsfunktion definiert:

  – Der eingegebene Text muß sich an folgende Syntax halten:

    activation_function_text ::= *“activation(“* **var_name** *“)“*
    “=“ **funct_expression.**

var_name	::=	“*x*“ | “*x_1*“.
funct_expression	::=	**term** { (“+“ | “–“) **term** }.
term	::=	**factor** { (“∗“ | “/“) **factor** }.
factor	::=	**power** { „ˆ “ **power** }.
power	::=	[“–“] **scalar**.
scalar	::=	“(“ **funct_expression** “)“ | **function**.
function	::=	“*sin(*“ **funct_expression** “*)*“ |
		“*cos(*“ **funct_expression** “*)*“ |
		“*tan(*“ **funct_expression** “*)*“ |
		“*asin(*“ **funct_expression** “*)*“ |
		“*acos(*“ **funct_expression** “*)*“ |
		“*atan(*“ **funct_expression** “*)*“ |
		“*sinh(*“ **funct_expression** “*)*“ |
		“*cosh(*“ **funct_expression** “*)*“ |
		“*tanh(*“ **funct_expression** “*)*“ |
		“*sqrt(*“ **funct_expression** “*)*“ |
		“*ln(*“ **funct_expression** “*)*“ |
		“*log(*“ **funct_expression** “*)*“ |
		var_or_const |
		own_function |
		var_name.
var_or_const	::=	**„Eine Konstante oder Variable“**
own_function	::=	**„Eine vom Benutzer definierte Funktion (vom Typ int oder double). Syntax: siehe Bemerkung.“**

Der Text muß dieselbe Syntax aufweisen wie der in Unterabschnitt 11.3.1 eingeführte *activation*–Befehl:

∗ **var_name** ist das Funktionsargument. Dieser Name ist nur hier lokal bekannt. D.h. es können beliebige — auch vordefinierte — Bezeichner verwendet werden; die ursprüngliche Bedeutung wird kurzfristig verdeckt!
Wenn im System eine vom Benutzer definierte Aktivierungsfunktion vorliegt, dann erscheint im Text die Variable *x_1*; ansonsten wird *x* verwendet. Der Anwender kann den Variablennamen aber beliebig ändern (sofern es ein gültiger Bezeichner ist).

∗ Bei der Definition können die beiden Datentypen *int* und *double* beliebig miteinander vermischt werden.

∗ Indem z.B. Operationen auf Konstanten zusammengefaßt werden, versucht ProSiNN **funct_expression** zu vereinfachen. Dabei werden aber Teilausdrücke nicht permutiert, so daß eine lokale und keine globale Vereinfachung stattfindet.

∗ Für **own_function** gilt die gleiche Syntax wie bei einer Funktionsauswertung (siehe Unterabschnitt 5.1.4); es darf aber zusätzlich noch das in der Definition vorkommende Funktionsargument verwendet werden!

∗ Durch den Button wird die neue Aktivierungsfunktion und deren Ableitung erzeugt:

· Die Aktivierungsfunktion wird mit dem Bezeichner **activation_funct** als Funktion vom Typ *double* generiert.

· Mit dem Bezeichner **der_activation_funct** wird die Ableitung der Aktivierungsfunktion gespeichert (vom Typ *double*).

Beide Funktionen können vom Benutzer nicht durch den *free_var*– oder den *free*–Befehl (siehe Abschnitt 13.4.2) gelöscht werden!
War der Bezeichner **activation_funct** oder **der_activation_funct** schon vorher im System definiert, dann wird die ursprünglich damit verbundene Variable gelöscht und mit der Aktivierungsfunktion bzw. der Ableitung neu definiert!

– Dieser Menüpunkt ist immer ausführbar! Es ist also nicht nötig, vorher ein Neuronales Netz im System zu definieren.

– Man beachte, daß eine eigene Aktivierungsfunktion den Verlauf des Netztrainings beeinflußt. Zum einen ist die Rechengenauigkeit nicht so hoch wie die der vordefinierten sigmoiden Funktion und zum anderen wird bei der Berechnung mehr Zeit benötigt.

– Paßt der eingegebene Text nicht mehr in das in Bild 17.3 dargestellte Texteingabefenster, so erzeugt ProSiNN ein vertikales und evtl. ein horizontales Scrollfeld. Mit ihnen kann der Benutzer jeden Teil des Textes betrachten und modifizieren.

– Entdeckt das System einen Fehler im Eingabetext, so erscheint der invertierte Fehlerausgabebutton (mit der Fehlermeldung), der über dem Ein/Ausgabefenster plaziert wird (siehe Unterabschnitt 3.4.2). Nachdem der Anwender den Button mit einer Maustaste angeklickt hat, ist die Fehlerstelle im Eingabefenster der Aktivierungsfunktion invertiert dargestellt.

– Lag im System schon eine neu definierte Aktivierungsfunktion vor, dann wird die entsprechende Funktion im Eingabetext dargestellt. Der Benutzer kann dann eine beliebige (erlaubte) Korrektur der Funktion durchführen.

17.3 Der Menüpunkt „Ableitung der Aktivierungsfunktion setzen“

Soll die Ableitung einer neu definierten Aktivierungsfunktion geändert werden, so muß der Menüpunkt *Ableitung der Aktivierungsfunktion setzen* ausgewählt werden. Anschließend erscheint z.B. das in Abbildung 17.4 dargestellte Popup–Fenster (die zugehörige Aktivierungsfunktion ist x).

Bem.:

- Durch Anklicken des Buttons [Abbruch] wird der Menüpunkt ohne Änderung der Aktivierungsfunktionsableitung beendet.

- Mit dem Button [Ableitung der Aktivierungsfunktion setzen] wird der darüberliegende Text als Ableitung der Aktivierungsfunktion definiert:

Abbildung 17.4: Das Popup–Fenster des Menüpunktes *Ableitung der Aktivierungsfunktion setzen*

- Der eingegebene Text muß sich an folgende Syntax halten:

  derivation_activation_function_text ::=
  “derivation_of_activation_funct(x_1) =“ **funct_expression.**

  Die Syntax der Ableitung der Aktivierungsfunktion ist dieselbe, wie die des in Unterabschnitt 11.3.2 eingeführten *derivation_of_activation_funct*–Befehls:

  * *x_1* ist das Funktionsargument. Der Anwender kann den Variablennamen aber beliebig ändern (sofern ein gültiger Bezeichner verwendet wird).
  * Für **funct_expression** gelten dieselben Angaben, die im letzten Abschnitt gemacht worden sind.
  * Durch den Button wird die Ableitung der Aktivierungsfunktion mit dem Bezeichner **der_activation_funct** als Funktion vom Typ *double* generiert.
- Dieser Menüpunkt ist nur ausführbar, wenn eine vom Benutzer definierte Aktivierungsfunktion im System vorliegt!
- Mit diesem Menüpunkt werden die folgenden positiven Aspekte für den Anwender angeboten:
  * Durch das *activation*–Kommando oder durch den im letzten Abschnitt beschriebenen Menüpunkt können bei der Ableitung singuläre Stellen auftreten (die real gar nicht vorhanden sein müssen). Mit diesem Menüpunkt lassen sich solche Definitionslücken entfernen!
  * Die Ableitung der Aktivierungsfunktion kann mit diesem Menüpunkt evtl. so vereinfacht werden, daß eine Beschleunigung des Lernverfahrens erzielt wird.
  * Der Benutzer kann Experimente mit verschiedenen Ableitungen der Aktivierungsfunktion durchführen.
- Paßt der eingegebene Text nicht mehr in das in Bild 17.4 dargestellte Texteingabefenster, so erzeugt ProSiNN ein vertikales und evtl. ein horizontales Scrollfeld. Mit ihnen kann der Anwender jeden Teil des Textes betrachten und modifizieren.
- Entdeckt das System einen Fehler im Eingabetext, so erscheint der invertierte Fehlerausgabebutton (mit der Fehlermeldung), der über dem Ein/Ausgabefenster plaziert wird (siehe Unterabschnitt 3.4.2). Nachdem der Benutzer den But-

ton mit einer Maustaste angeklickt hat, ist die Fehlerstelle im Eingabefenster der Aktivierungsfunktionsableitung invertiert dargestellt.

17.4 Der Menüpunkt „Normale (sigmoide) Aktivierungsfunktion setzen“

Will der Benutzer wieder die sigmoide Aktivierungsfunktion setzen, so kann er den Menüpunkt *Normale (sigmoide) Aktivierungsfunktion setzen* anwählen. Anschließend erscheint das Popup–Fenster, das in Abbildung 17.5 dargestellt ist:

Abbildung 17.5: Das Popup–Fenster des Menüpunktes *Normale (sigmoide) Aktivierungsfunktion setzen*

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt ohne Veränderung der Aktivierungsfunktion beendet.
- Durch Auswahl des Buttons Setze 'normale' Aktivierungsfunktion wird wieder auf die intern implementierte sigmoide Aktivierungsfunktion umgeschaltet.

  Diese Vorgehensweise ist identisch mit dem Ausführen des *normal_activation*–Befehls in Unterabschnitt 11.3.3.

17.5 Der Menüpunkt „Ausgabefenster speichern“

Soll der Inhalt des Ein/Ausgabefensters in einer Datei gespeichert und evtl. anschließend noch gelöscht werden, so muß man den Menüpunkt *Ausgabefenster speichern* anwählen. Danach erscheint das in Bild 17.6 dargestellte Popup–Fenster:

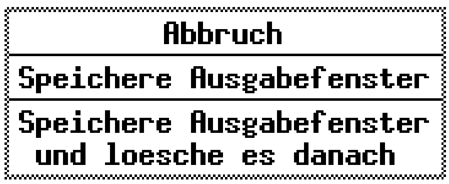


Abbildung 17.6: Das Popup–Fenster des Menüpunktes *Ausgabefenster speichern*

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt ohne Speicherung des Ein/Ausgabefensters beendet.

- Durch Auswahl des Buttons Speichere Ausgabefenster erscheint eine Dateiauswahlbox (siehe Unterabschnitt 3.4.2), die das Sichern des Ein/Ausgabefensterinhalts in einer Datei ermöglicht.

- Durch Anklicken des Buttons Speichere Ausgabefenster und loesche es danach erscheint eine Dateiauswahlbox (siehe Unterabschnitt 3.4.2), die das Speichern des Inhaltes des Ein/Ausgabefensters in einer Datei erlaubt. Anschließend wird der Inhalt des Fensters gelöscht.

17.6 Der Menüpunkt „Ausgabefenster loeschen“

Will der Anwender den Inhalt des Ein/Ausgabefensters löschen, so muß er den Menüpunkt *Ausgabefenster loeschen* anwählen. Anschließend erscheint das Popup–Fenster, das in Abbildung 17.7 dargestellt ist:

Abbildung 17.7: Das Popup–Fenster des Menüpunktes *Ausgabefenster loeschen*

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt ohne das Löschen des Ein/Ausgabefensterinhalts beendet.

- Durch Auswahl des Buttons Ausgabefenster loeschen wird der Inhalt des Ein/Ausgabefensters gelöscht.

17.7 Der Menüpunkt „Alles freigeben“

Um das System wieder in den Zustand zu versetzen, der nach dem Systemstart vorlag, kann der Menüpunkt *Alles freigeben* angewählt werden. Danach erscheint die Auswahlbox, die in Bild 17.8 dargestellt ist:

Abbildung 17.8: Das Popup–Fenster des Menüpunktes *Alles freigeben*

Bem.:

- Durch Anklicken des Buttons Abbruch wird der Menüpunkt abgebrochen.

- Durch den Button $\boxed{\text{Alles freigeben}}$ wird folgendes ausgeführt:
  - Alle mit der *define*–Anweisung erzeugten Variablen werden wieder freigegeben. Dazu gehören Integer–, Double–, Feld–, Funktions–, String–, Layer–, Unit– und Weight–Variablen.
  - Alle durch *alias* generierten Makros werden aus dem System entfernt.
  - Ein existierendes Neuronales Netz wird eliminiert.
  - Testmuster und Trainingsmuster werden gelöscht.
  - Es wird wieder die sigmoide Aktivierungsfunktion gesetzt.
  - Alle Dateien und Grafikfenster werden geschlossen.
  - Alle Text– und Grafikprotokolle werden gelöscht.
  - Der Variablen **net_name** wird der leere String zugewiesen.
  - Die folgenden vordefinierten Variablen werden auf die angegebenen Werte gesetzt:
    * **precision** = 8
    * **floatpattern** = **on**
    * **pat_min** = 0.0 und **pat_max** = 1.0
    * **theta_min** = −0.5 und **theta_max** = 0.5
    * **w_min** = −0.5 und **w_max** = 0.5
    * **act_error** = **old_error** = 0.0
    * **error_limit** = 0.001
    * **error_mode** = **quadratisch**
    * **eta** = **act_eta** = 0.2
    * **alpha** = **act_alpha** = 0.9
    * **fail_limit** = 1.04
    * **phi** = 1.05
    * **beta** = 0.7
    * **sum_iterations** = **act_iterations** = 0
  - Nach der Ausführung existieren *keine* vom Benutzer definierten Bezeichner mehr im System!

Kapitel 18

Das Menü „Benutzerwerte bearbeiten“

Dieses Kapitel beschäftigt sich mit dem Menü *Benutzerwerte bearbeiten.* Hier werden sieben Menüpunkte angeboten, die folgendes ermöglichen:

- Zum Bearbeiten von Dateien läßt sich ein Editor aufrufen
- Ein ProSiNN–Programm kann geladen und ausgeführt werden
- Die folgenden Variablen können erzeugt werden:
  - Integer und Double
  - String
  - Integer– und Doublefelder
  - Integer– und Doublefunktionen
  - Kommandomakros
  - Existiert ein Netz im System, dann können außerdem noch Netzvariablen generiert werden
- Die nachfolgenden Variablen können bearbeitet oder freigegeben werden:
  - Integer und Double
  - String
  - Integer– und Doublefelder
  - Integer– und Doublefunktionen:
    - Funktionswerte können berechnet werden
    - Eine Funktion kann abgeleitet werden
    - Funktionen können modifiziert werden
    - Eine Funktion kann in ein Grafikfenster ausgegeben werden
  - Kommandomakros

  - Netzvariablen

- Grafikfenster können bearbeitet werden:
  - Neue Grafikfenster lassen sich erzeugen
  - In einem Grafikfenster können die folgenden Operationen ausgeführt werden:
    * Das Grafikfenster kann eliminiert werden
    * Der Inhalt kann gelöscht werden
    * Das Weltkoordinatensystem kann gesetzt werden
    * Koordinatenachsen können eingezeichnet werden
    * Eine Linie kann (gestrichelt) erzeugt werden
    * Ein Text kann ausgegeben werden
  - Alle Grafikfenster können eliminiert werden

- Alle vom Anwender definierten Variablen können wieder freigegeben werden

Die Abbildung 18.1 zeigt das Menü in der Form, die der Benutzer nach dem Systemstart erhält:

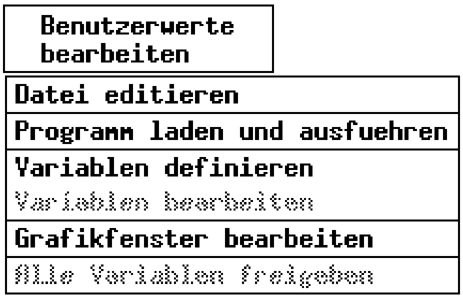


Abbildung 18.1: Das Menü *Benutzerwerte bearbeiten*

Die folgenden Abschnitte führen die einzelnen Menüpunkte ein.

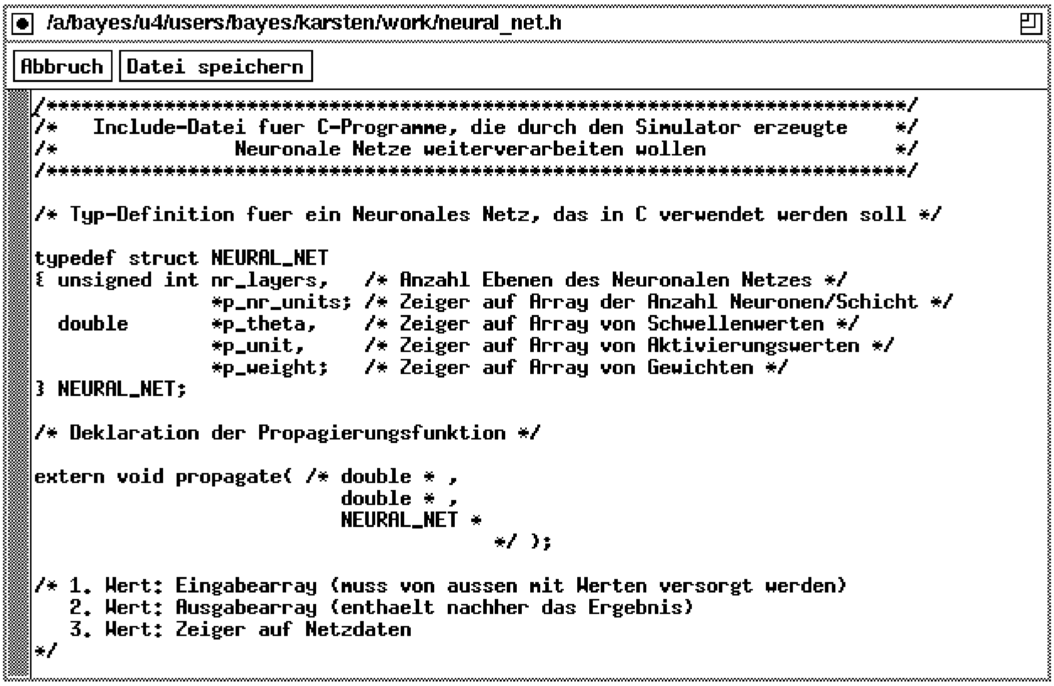
18.1 Der Menüpunkt „Datei editieren“

Will der Benutzer eine neue Datei kreieren oder eine existierende modifizieren, so muß er den Menüpunkt *Datei editieren* anwählen. Nach dessen Auswahl erscheint eine Dateiauswahlbox (siehe Unterabschnitt 3.4.2), die das Erzeugen oder Laden einer Datei ermöglicht.

Bem.:

- Es können nur solche Dateien editiert werden, auf die der Anwender auch Schreibrechte hat.

- Nachdem eine Datei zum Editieren ausgewählt wurde, wird ein neues Fenster erzeugt, in dem die Datei bearbeitet werden kann. In Bild 18.2 ist das Editorfenster dargestellt (hier wurde die *include*–Datei `neural_net.h`, die in Unterabschnitt 9.5.1 beschrieben wurde, ausgewählt):




```
/************************************************************************/
/*   Include-Datei fuer C-Programme, die durch den Simulator erzeugte    */
/*              Neuronale Netze weiterverarbeiten wollen                  */
/************************************************************************/

/* Typ-Definition fuer ein Neuronales Netz, das in C verwendet werden soll */

typedef struct NEURAL_NET
{ unsigned int nr_layers,   /* Anzahl Ebenen des Neuronalen Netzes */
               *p_nr_units; /* Zeiger auf Array der Anzahl Neuronen/Schicht */
  double       *p_theta,    /* Zeiger auf Array von Schwellenwerten */
               *p_unit,     /* Zeiger auf Array von Aktivierungswerten */
               *p_weight;   /* Zeiger auf Array von Gewichten */
} NEURAL_NET;

/* Deklaration der Propagierungsfunktion */

extern void propagate( /* double * ,
                          double * ,
                          NEURAL_NET *
                                       */ );

/* 1. Wert: Eingabearray (muss von aussen mit Werten versorgt werden)
   2. Wert: Ausgabearray (enthaelt nachher das Ergebnis)
   3. Wert: Zeiger auf Netzdaten
*/
```

Abbildung 18.2: Das Editorfenster des Menüpunktes *Datei editieren*

– Die Größe des Fensters korrespondiert mit der Größe des Ein/Ausgabefensters (sie kann manuell verändert werden). Paßt der eingegebene Text nicht (mehr) in das Fenster, so erzeugt ProSiNN horizontale und/oder vertikale Scrollfelder. Mit ihnen kann der Benutzer jeden Teil des Textes erreichen.

– Wenn die Datei noch nicht existierte, dann ist der Inhalt des Editorfensters leer.

– Das Fenster ist (relativ) unabhängig vom übrigen System. Nur wenn ProSiNN auf eine Eingabe wartet, kann der Anwender im Editorfenster keine Änderungen durchführen. Wird das System beendet, so wird auch das Editorfenster ohne Speichern der Datei abgebrochen.

– Der Fenstername bezeichnet den Pfad und den Namen der Datei, die editiert werden soll (`/a/bayes/u4/users/bayes/karsten/work/neural_net.h` ist der Fenstername in Abbildung 18.2).

– Zur Textbearbeitung können die Maustasten und Tastaturkombinationen, die in Unterabschnitt 3.4.1 beschrieben wurden, verwendet werden.

– Die beiden Buttons [Abbruch] und [Datei speichern] dienen zum Beenden des Editors und zum Speichern des Textes:

  * Nachdem der Button [Abbruch] angewählt wurde, erscheint die in Abbildung 18.3 dargestellte Auswahlbox.

    · Mit dem Button [Abbruch (weiter editieren)] kann der Anwender die Auswahlbox abbrechen und den vorliegenden Text weiter editieren.

    · Mit dem Button [Editor verlassen] wird das Editorfenster ohne Speichern des Textes beendet.

Abbildung 18.3: Die Auswahlbox des Editorfensters

* Will der Benutzer den Text in der angegebenen Datei speichern, so muß er mit dem Mauszeiger den Button Datei speichern anwählen. Sollte beim Schreiben ein Fehler auftreten, so wird dies vom System gemeldet.

18.2 Der Menüpunkt „Programm laden und ausfuehren“

Mit dem Menüpunkt *Programm laden und ausfuehren* kann ein ProSiNN–Programm ausgeführt werden.

Bem.:

- Nachdem der Menüpunkt angewählt wurde, erscheint eine Dateiauswahlbox (siehe Unterabschnitt 3.4.2), die das Laden und Ausführen eines Programms ermöglicht.

- Der syntaktische Aufbau eines Simulationsprogramms muß den folgenden Bedingungen genügen:

  - Das auszuführende Programm besteht aus einem **statement_block** (siehe Unterabschnitt 6.1.2) beliebiger Größe. Jedes Kommando muß mit einem Semikolon abgeschlossen werden. Innerhalb dieser Anweisungen dürfen auch *load*–Befehle auftreten. So kann z.B. ein Programm in Abhängigkeit einer Bedingung geladen und ausgeführt werden.
  - Zur Dokumentation des Programms können Kommentare verwendet werden. Ein Kommentar ist folgendermaßen definiert:

    comment ::= “/*“ { **char_const** } “*/“.

    In einem Kommentar dürfen sich weitere Kommentare befinden. Damit ist es möglich, während eines Testlaufs kommentierte Befehlssequenzen von der Interpretierung auszuschließen.

- Zur vorzeitigen Beendigung eines Programms kann der Benutzer die Ausführung abbrechen lassen. Dazu muß der Mauszeiger in das Ein/Ausgabefenster gesetzt und die Tastenkombination `Control-C` gedrückt werden.

- Von einem Programm generierte Variablen sind nach dessen Ausführung noch im System vorhanden — sofern sie nicht explizit wieder freigegeben werden. Das Simulationsprogramm selbst wird aber aus dem Speicher entfernt. Um es erneut zu starten, muß der Menüpunkt wieder aufgerufen oder das entsprechende *load*–Kommando eingegeben werden.

18.3 Der Menüpunkt „Variablen definieren“

Will der Benutzer Variablen oder Kommandosmakros erzeugen, so kann er den Menüpunkt *Variablen definieren* anwählen. Danach erscheint das Popup–Fenster, das in Abbildung 18.4 dargestellt ist:

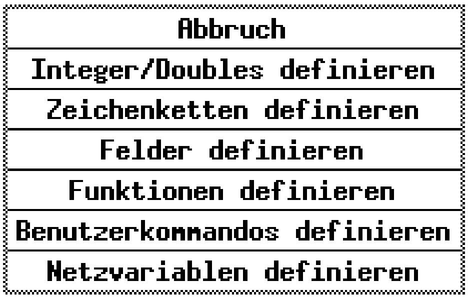


Abbildung 18.4: Die Auswahlbox des Menüpunktes *Variablen definieren*

Bem.:

- Mit dem Button Abbruch kann der Menüpunkt beendet werden.
- Die Verwendung der anderen Buttons wird in den folgenden Unterabschnitten beschrieben.

18.3.1 Der Menüunterpunkt „Integer/Doubles definieren“

Mit dem Button Integer/Doubles definieren kann der Anwender Variablen vom Typ *int* oder *double* generieren und beliebig initialisieren. Nach dessen Auswahl erscheint das Popup–Fenster, das in Bild 18.5 dargestellt ist:

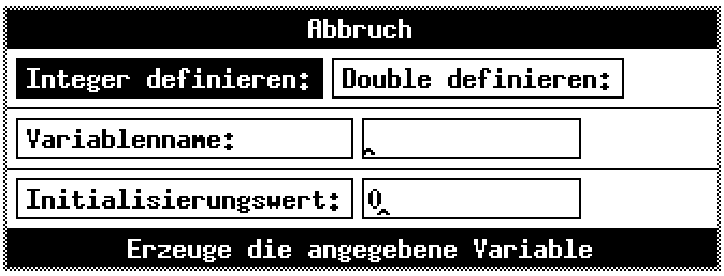


Abbildung 18.5: Das Popup–Fenster des Menüunterpunktes *Integer/Doubles definieren*

Bem.:

- Durch Anklicken des Buttons Abbruch wird das Fenster beendet und die Auswahlbox, die in Abbildung 18.4 dargestellt ist, erscheint wieder.

- Die beiden Buttons $\boxed{\text{Integer definieren}}$ und $\boxed{\text{Double definieren}}$ dienen als Umschalter (Toggles) zwischen den beiden möglichen Variablentypen. Wenn der Toggle $\boxed{\text{Integer definieren}}$ aktiv ist (wie in Bild 18.5), dann kann der Benutzer eine Integervariable erzeugen. Ist dagegen $\boxed{\text{Double definieren}}$ aktiv, so wird eine Gleitkommavariable generiert. Die Umschaltung zwischen den beiden Alternativen wird folgendermaßen durchgeführt:
  - Den Mauszeiger auf den inaktiven Button setzen.
  - Durch Drücken der linken Maustaste wird der gewünschte Variablentyp gesetzt.
- Der Bezeichner, den die zu erzeugende Variable erhalten soll, wird wie folgt eingegeben:
  - Den Mauszeiger auf das Eingabefeld für den Variablennamen setzen.
  - Mit der Tastatur einen gültigen (siehe Abschnitt 4.1) noch nicht definierten Bezeichner eingeben.
- Der Wert, mit dem die Variable initialisiert werden soll, wird wie anschließend beschrieben eingegeben:
  - Den Mauszeiger auf das Eingabefeld für den Initialisierungswert setzen.
  - Den dort angegebenen Wert auf die gewünschte Zahl korrigieren, die mit dem gesetzten Variablentyp konform sein muß.
- Nachdem der Anwender alle benötigten Eingaben vorgenommen hat, kann die Variable durch Anklicken des Buttons $\boxed{\text{Erzeuge die angegebene Variable}}$ definiert werden.

  Wurde die Variable korrekt erzeugt, so erscheint wieder die Auswahlbox, die in Abbildung 18.4 dargestellt ist.

  Konnte die Variable nicht generiert werden, so wird eine Fehlermeldung ausgegeben und der Benutzer kann seine Eingaben korrigieren.

18.3.2 Der Menüunterpunkt „Zeichenketten definieren“

Mit dem Button $\boxed{\text{Zeichenketten definieren}}$ kann der Anwender eine Variable vom Typ *string* erzeugen und beliebig initialisieren. Nach dessen Auswahl erscheint das Popup–Fenster, das in Bild 18.6 dargestellt ist.

Bem.:

- Durch Anklicken des Buttons $\boxed{\text{Abbruch}}$ wird das Fenster beendet und die Auswahlbox, die in Abbildung 18.4 dargestellt ist, erscheint wieder.
- Der Bezeichner, den die zu erzeugende Variable erhalten soll, wird wie folgt eingegeben:
  - Den Mauszeiger auf das Eingabefeld für den Variablennamen setzen.
  - Mit der Tastatur einen gültigen (siehe Abschnitt 4.1) noch nicht definierten Bezeichner eingeben.

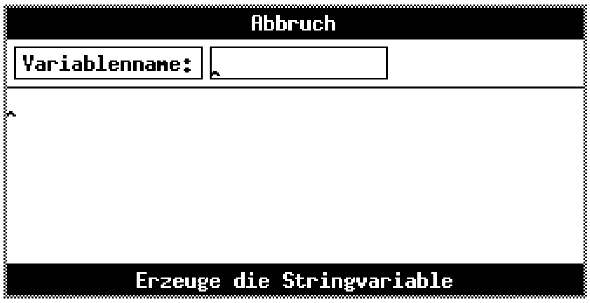


Abbildung 18.6: Das Popup–Fenster des Menüunterpunktes *Zeichenketten definieren*

- Die Zeichenkette, mit der die Variable initialisiert werden soll, wird folgendermaßen eingegeben:
  - Den Mauszeiger auf das Eingabefeld für den Text setzen.
  - Jetzt kann eine beliebige Zeichenkette eingegeben werden (auch das Zeilenendezeichen und der leere String sind erlaubt).
  - Paßt der Text nicht mehr in das in Bild 18.6 dargestellte Eingabefenster, so erzeugt ProSiNN ein horizontales und/oder ein vertikales Scrollfeld. Mit ihnen kann jeder Teil der Zeichenkette erreicht werden.
- Nachdem der Anwender alle benötigten Eingaben vorgenommen hat, kann die Variable durch Anklicken des Buttons Erzeuge die Stringvariable definiert werden.

  Wurde die Variable korrekt erzeugt, so erscheint wieder die Auswahlbox, die in Abbildung 18.4 dargestellt ist.

  Konnte die Variable nicht generiert werden, so wird eine Fehlermeldung ausgegeben und der Benutzer kann seine Eingaben korrigieren.

18.3.3 Der Menüunterpunkt „Felder definieren“

Mit dem Button Felder definieren kann der Anwender eine Feldvariable vom Typ *int* oder *double* erzeugen und beliebig initialisieren. Nach dessen Auswahl erscheint das Popup–Fenster, das in Bild 18.7 dargestellt ist.

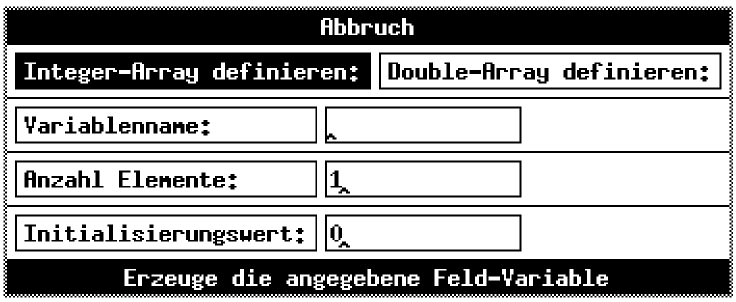


Abbildung 18.7: Das Popup–Fenster des Menüunterpunktes *Felder definieren*

Bem.:

- Durch Anklicken des Buttons $\boxed{\text{Abbruch}}$ wird das Fenster beendet und die Auswahlbox, die in Abbildung 18.4 dargestellt ist, erscheint wieder.
- Die beiden Buttons $\boxed{\text{Integer–Array definieren}}$ und $\boxed{\text{Double–Array definieren}}$ dienen als Umschalter (Toggles) zwischen den beiden möglichen Variablentypen. Wenn der Toggle $\boxed{\text{Integer–Array definieren}}$ aktiv ist (wie in Bild 18.7), dann kann der Benutzer eine Integerfeldvariable erzeugen. Ist dagegen $\boxed{\text{Double–Array definieren}}$ aktiv, so wird eine Gleitkommafeldvariable generiert. Die Umschaltung zwischen den beiden Alternativen wird folgendermaßen durchgeführt:
  - Den Mauszeiger auf den inaktiven Button setzen.
  - Durch Drücken der linken Maustaste wird der gewünschte Variablentyp gesetzt.
- Der Bezeichner, den die zu erzeugende Variable erhalten soll, wird wie folgt eingegeben:
  - Den Mauszeiger auf das Eingabefeld für den Variablennamen setzen.
  - Mit der Tastatur einen gültigen (siehe Abschnitt 4.1) noch nicht definierten Bezeichner eingeben.
- Die Anzahl der Feldelemente wird wie nachfolgend beschrieben eingegeben:
  - Den Mauszeiger auf das Eingabefeld für die Anzahl der Elemente setzen.
  - Die dort angegebene ganze Zahl auf den gewünschten Wert korrigieren (die Elementanzahl muß größer als 0 sein).
- Der Wert, mit dem alle Feldelemente initialisiert werden sollen, wird folgendermaßen eingegeben:
  - Den Mauszeiger auf das Eingabefeld für den Initialisierungswert setzen.
  - Den dort angegebenen Wert auf die gewünschte Zahl korrigieren, die mit dem gesetzten Variablentyp konform sein muß.
- Nachdem der Anwender alle benötigten Eingaben vorgenommen hat, kann die Variable durch Anklicken des Buttons $\boxed{\text{Erzeuge die angegebene Feld–Variable}}$ definiert werden.

  Wurde die Variable korrekt erzeugt, so erscheint wieder die Auswahlbox, die in Abbildung 18.4 dargestellt ist.

  Konnte die Variable nicht generiert werden, so wird eine Fehlermeldung ausgegeben und der Benutzer kann seine Eingaben korrigieren.

18.3.4 Der Menüunterpunkt „Funktionen definieren“

Mit dem Button $\boxed{\text{Funktionen definieren}}$ kann der Anwender eine Funktionsvariable vom Typ *int* oder *double* erzeugen. Nach dessen Auswahl erscheint das Popup–Fenster, das in Bild 18.8 dargestellt ist: