PR: Programmierpraktikum II (SS2000)

Visualization of Artificial Neuronal Networks

Andreas Lackner
luke@cosy.sbg.ac.at

Marko Jestratijevic
mjestrat@cosy.sbg.ac.at

Version 2000-25-10

Gruppe 2
Zeit: Mittwoch 16.30 - 1700
Lehrveranstaltungsleiter: Schwaiger Roland

Inhaltsverzeichnis

Abbildungsverzeichnis

ABSTRACT

Das Programmpaket dient zum Betrachten und Verändern von Neuronalen Netzen, die im SNNS-Format gespeichert sind. Der NetExplorer übernimmt dabei die Aufgabe des Filebrowser, der ein einfaches Navigieren im Verzeichnisbaum ermöglicht und vorhandene SNNS-Dateien als kleine Grafik darstellt (Thumbnails), und bei Bedarf das Programm NetViewer startet. Der NetViewer stellt das Neuronale Netz graphisch dar und ermöglicht die einfache Änderung der verschiedenen Darstellungsparameter wie Farben, Formen oder Positionen einzelner Elemente. Die Änderungen können jederzeit gespeichert werden und als PostScript exportiert werden.

Kapitel 1

Einleitung

1.1 Einführung zum Programm

Dieses Programm wurde mit der Absicht hergestellt, die graphische Arbeit mit Neuronalen Netzen zu ermöglichen und zu erleichtern. Das Programm hat die Möglichkeit die als SNNS-Dateien gespeicherten Daten einzulesen. Nach der Analyse können die Daten angezeigt und bearbeitet werden. Weiterhin kann man mit dem Programm den Verzeichnisbaum auf SNNS-Dateien durchsuchen. Der graphische Aufbau der Netze kann nach eigenen Wünschen und Bedürfnissen geändert und angepaßt werden. Alle ausgeführten Veränderungen können gespeichert werden. Dem Benutzer sind auch alle anderen mit dem Neuronalen Netz zusammenhängenden Daten verfügbar. In einer zweiten Ausbau-Stufe des Programmpakets soll die Möglichkeit gegeben sein, auch diese Daten zu verändern und neue Knoten und Verbindungen in ein Netz einzufügen. Damit würde das Programmpaket alle Möglichkeiten bieten um Netze ohne der Hilfe des SNNS-Pakets aufzubauen.Das Programm besitzt verschiedene graphische Werkzeuge, welche dem Benutzer die Arbeit erleichtern und darüberhinaus die Darstellung der Netze verschönern. Dabei denkt man an die Möglichkeiten der Benutzung vieler Farben, die Veränderung des Knoten-Aussehens und dem Einfügen von Titeln und Notizen. Die graphische Darstellung wird, sofern dafür keine Datenfelder in der SNNS-Datei vorgesehen sind, in einem eigenen File abgelegt, weiters kann man die Daten als PostScript exportieren.

Es muß erwähnt werden, daß für die Realisation dieses Programms auch bestimmte Softwarepakete (GifEncoder, ImageEncoder ,IntHashtable ,PSGr) von anderen Autoren eingebunden wurden, welchen wir auf diesem Weg danken. Da anzunehmen ist, daß diese Pakete zum Standard einer modernen Programmiersprache gehören, stehen sie nicht im Klassendiagramm.

Während der Arbeit und der Entwicklung dieses Programms wurden zusätzliche Möglichkeiten bemerkt, welche das Programm besser und die Arbeit damit benutzerfreundlicher machen würden. Diese Funktionen werden in der eventuellen zweiten Version dieses Programms eingearbeitet. Soweit zur Einleitung und zu den Eckdaten des Programms. Etwas mehr über Neuronale Netze und Details über das Aussehen und die Möglichkeiten des Programms werden im nächsten Kapiteln beschrieben.

1.2 Kurze Beschreibung von ANNs

Der folgende Paragraph beschreibt das Modell für Neuronale Netze,die vom SNNS-Paket generiert werden. Die Netz besteht aus Knoten (Units) und Verbindungen (Connections), die zwischen jeweils zwei Knoten liegen.

- KNOTEN: Es gibt drei Typen von Knoten mit ihren verschiedenen Funktionen im Netz.Input-Units(die Units deren Aktivierung den Eingang ins Netz darstellen, Output-Units (deren Ausgang zu keiner anderen Unit verbunden ist) und die restlichen Units sind sogenannte verstecke (hidden) Units.Jedes Unit hat die folgende Attributen, die in einer Tabelle im der SNNS-Datei gespeichert wird:
  - no: fortlaufende Nummer, um die Units zu unterscheiden (key)
  - name: vom Benutzer frei wählbarer Name, der der Unit zugewiesen wird
  - io-type: Art der Unit (input / output / hidden usw.). Der Typ bestimmt die Funktion des Units im Netz.
  - activation: Aktivierungs Wert der Unit
  - a-act: enthält initial-activation Wert
  - bias: bestimmt wo die activation function den steilsten Aufstieg hat
  - actFunc: activation function
  - outFunc: output function
  - f-type: Funktions-Typen und Prototypen
  - position: Die Koordinaten im Raum für graphische Darstellung (X,Y,Z)
  - subnet-no: Mit der Benutzung dieser variablen Struktur der Netze kann viel klarer dargestellt werden,als in einer2D-Präsentation möglich wäre
  - frozen: Boolescher Wert der angibt ob activation und output geändert werden können während der Simulation
- VERBINDUNGEN: eine Verbindung verbindet immer zwei Knoten:Source Unit und target Unit, dabei ist es möglich daß die source und target Unit identisch sind.Jede Verbindung hat ein Gewicht, die die Wirkung der source Unit auf die target Unit auf folgende Weise definiert: Ist das Gewicht negativ, dann ist die Verbindung inhibitorisch, d.h. es vergrößert die Wirkung auf die target Unit,

wenn es positiv ist, dann hat es einen exzitatorischen Effekt,bzw. es vermindert die Wirkung.

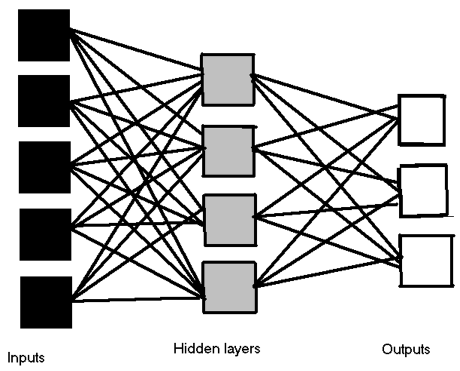

Abbildung 1.1: Beispiel eines einfachen Feed-Forward Netzes

1.3 Darstellungsmöglichkeiten von Neuronalen Netzen

Künstliche Neuronale Netze (ANNs) bestehen aus Knoten und Verbindungen. Eine Verbindung hat einen Start- und einen Endknoten, die ident sein können. Damit ist die graphische Darstellung logisch festgelegt. Was jedoch nicht feststeht ist die topographische Lage der Knoten in ihrer Umgebung. Bei der Erstellung von ANNs mit Hilfe des SNNS-Pakets muß die Position eines Knotens im dreidimensionalen Raum vom Anwender eingegeben werden, um graphisches Arbeiten zu ermöglichen.

In der Literatur haben sich verschiedene Darstellungsformen für Netze durchgesetzt (zumindest in Bezug auf die Topographische Darstellung). Die graphische Darstellung eines Knotens ist meist als Quadrat oder Kreis angezeigt (bzw. Würfel oder Kugel im dreidimensionalen Darstellungen). Eine Verbindung wird als Linie vom Start- zum Endknoten gezeichnet.

1.4 Anforderungen an graphische Darstellungen

Grundsätzlich sollen Knoten so angeordnet werden, wie es den allgemeinen Standards der Literatur entspricht. In Abbildung 1.1 ist die typische Anordnung der Knoten eines Feed-Forward Netzes zu sehen. Dabei wird das Netz normalerweise in funktionell zusammen gehörige Ebenen oder Gruppen gegliedert, wobei diese wiederum von links nach rechts bzw. von oben nach unten angeordnet werden. Bei der Anordnung der

Knoten ist weiters zu beachten, daß die Verbindungslinien keine Knoten schneiden, um die “Lesbarkeit” des Bildes nicht zu beeinträchtigen. Texte, die Daten von Knoten beinhalten, müssen so plaziert werden, daß klar erkennbar ist welcher Text zu welchem Knoten gehört. Für jeden Knoten und jede Verbindung können verschiedene Daten interessant sein (Lage,Gewicht,Nummer...), es sollte daher für jeden Knoten einzeln einstellbar sein, welche Daten an welcher Stelle angezeigt werden.Die verschiedenen Daten über Knoten und Verbindungen wurden im Abschnitt 1.2 beschrieben. Werden zu viele Daten angezeigt leidet die Lesbarkeit, weiters ist es sowieso kaum möglich auf engstem Raum alle Daten darzustellen. Es muß also auch eine Weg gefunden werden, der es dem Anwender ermöglicht schnell und effizient genau die Daten anzuzeigen, die gerade interessant sind. Betreffend des Orts, an dem die Daten plaziert werden, sollte jeder seinen eigenen Vorlieben folgen können. Das bedeutet, das jedes Feld (in dem Daten oder Knoten dargestellt werden) beliebig verschiebbar sein soll. Da die Datenfelder in Bezug zu einem Knoten stehen, sollen die Datenfelder mitverschoben werden, wenn der Knoten verschoben wird. Diese Funktion kann unter Umständen stören, z.B. wenn der Anwender Datenfelder am Rand der Graphik anordnet, in solchen Fällen kann die Verschiebung der Datenfelder unterdrückt werden.

Kapitel 2

Use Cases

Im folgenden Kapitel sollen die verschieden Use-Cases beschrieben werden, wie sie in Abbildung A.1 auf Seite 19 dargestellt sind.

2.1 Verzeichnis auswählen

In einem Hauptfenster, das ähnlich einem Filebrowser aufgebaut ist, wählt der Anwender ein Verzeichnis aus einem Verzeichnisbaum durch Anklicken mit der Maustaste. Die SNNS Dateien haben die Endung .net und die Unterverzeichnisse des gewählten Verzeichnisses werden in einem eigenen Teil des Hauptfensters aufgelistet. Zu jedem *.net-File wird ein Thumbnail erzeugt, das die Struktur des Netzes darstellt. Die Daten dazu werden aus dem *.net-File gelesen.

2.2 Datei öffnen

Durch Doppelklicken mit der linken Maustaste auf eines der Thumbnails wird eine Datei geöffnet. Falls zur *.net Datei eine gleichnamige *.net.props-Datei existiert, wird auch dies geöffnet. In *.net-props-File sind die Daten gespeichert, die sich auf die graphische Darstellung beziehen und nicht im SNNS-File abgespeichert werden können, wie zum Beispiel Farben, Form, Hintergundfarbe und ähnliches. Falls diese Datei nicht im selben Verzeichnis existiert, wird eine solche Datei erzeugt. Die Daten beider Files werden später verwendet um eine entsprechende Darstellung zu erzeugen, die in einem eigenen Fenster dargestellt werden (Graphikfenster).

2.3 Netz zeichnen

Die in *.net -Files abgelegte topographische Anordnung der Units und Verbindungen eines Netzes wird verwendet um eine vereinfachte Übersichtsdarstellung des Netzes zu generieren, das als *.gif Datei abgelegt wird. Die Darstellung im Graphikfenster hängt aber auch von den Daten im *.net.props File ab. Um eine schnelle Möglichkeit anzubieten die Darstellung zu ändern, wird beim Zeichnen der Units auch deren Aktivierungs-Zustand beachtet (s.a. Unit aktivieren)

2.4 Unit aktivieren

Der Anwender aktiviert im Graphikfenster einen Knoten durch Anklicken mit der linken Maustaste. Das Anklicken eines bereits aktivierten Unit deaktiviert sie wieder. Der Aktivierungszustand wird im *.net-props gespeichert. Weiters besteht die Möglichkeit alle Knoten zu aktivieren oder den Aktivierungszustand aller Knoten umzukehren.

2.5 Unit-Daten in extra Fenster darstellen

Der Anwender öffnet im Graphikfenster einen Knoten durch Doppelklicken mit der linken Maustaste. Dadurch öffnet sich ein extra Fenster, in dem sämtliche zur Verfügung stehenden Daten, die diesen Knoten betreffen, dargestellt werden.

2.6 Datenfeld verschieben

Alle Datenfelder (also Textfelder mit Namen, Aktivierungszustand usw.) können vom Anwender im Graphikfenster verschoben werden (Drag & Drop).

2.7 Unit verschieben

Verschiebt der Anwender einen Knoten (s.a. Datenfeld verschieben), so hängt es von den eingestellten Eigenschaften des Knotens ab, ob die Datenfelder mitverschoben werden oder nicht (sticky-bit).

2.8 Datei speichern

Der Anwender kann die Daten durch Anwählen eines Menüpunkts abspeichern. Dabei werden Koordinaten der Knoten und die Struktur des Netzes im *.net-File gespeichert. Alle weiteren Daten werden im *.net-props File gespeichert. Der Anwender wird nach dem Verzeichnis und Namen der Datei gefragt, unter der er das Netz abspeichern will und gewarnt, falls er bestehende Dateien über-schreibt.

2.9 Screenshot exportieren

Die Darstellung des Netzes kann jederzeit auf *.PS Files exportiert werden, indem der Anwender einen entsprechenden Menüpunkt im Graphikfenster anwählt. Das Speichern der Graphik erfolgt durch einen eigenen Export-Filter.Je nach Vorhandensein der entsprechenden Export-Filter können auch verschieden Datei-Typen erzeugt werden.

Kapitel 3

NetExplorer

3.1 Beschreibung

Die Java-Applikation NetExplorer umfaßt die Use-Cases “Verzeichnis auswählen” und “Datei öffnen”. Das Hauptfenster ist in drei Teile geteilt, links wird ein Verzeichnisbaum gezeigt, rechts die im angewählten Verzeichnis vorhandenen SNNS-Netzdateien. In der Fußzeile des Fensters werden zusätzliche Daten, wie zum Beispiel der Fortschritt eines Ladevorgangs, angezeigt. Der Verzeichnisbaum hat als root das home-Verzeichnis des Anwenders.

3.2 Implementierte Funktionen

- Verzeichnisbaum darstellen
- Status-bar zeigen
- Thumbnails darstellen
- Aufruf des NetViewers als eigenen Thread.
- Status-Bar beim Thumbnails laden
- ToolTips bei Thumbnail-Namen
- Mit dem Auswahl eines SNNS-File in Verzeichnisbaum wird das entsprechende Thumbnail markiert

3.3 ToDo

- Icons im Verzeichnisbaum ändern.

- Verzeichnisbaum aktualisieren, wenn sich Verzeichnisse ändern.
- Startverzeichnis des Baums vom Anwender veränderbar machen.
- Drag & Drop zwischen NetExplorer und NetViewer

3.4 Class Diagrams

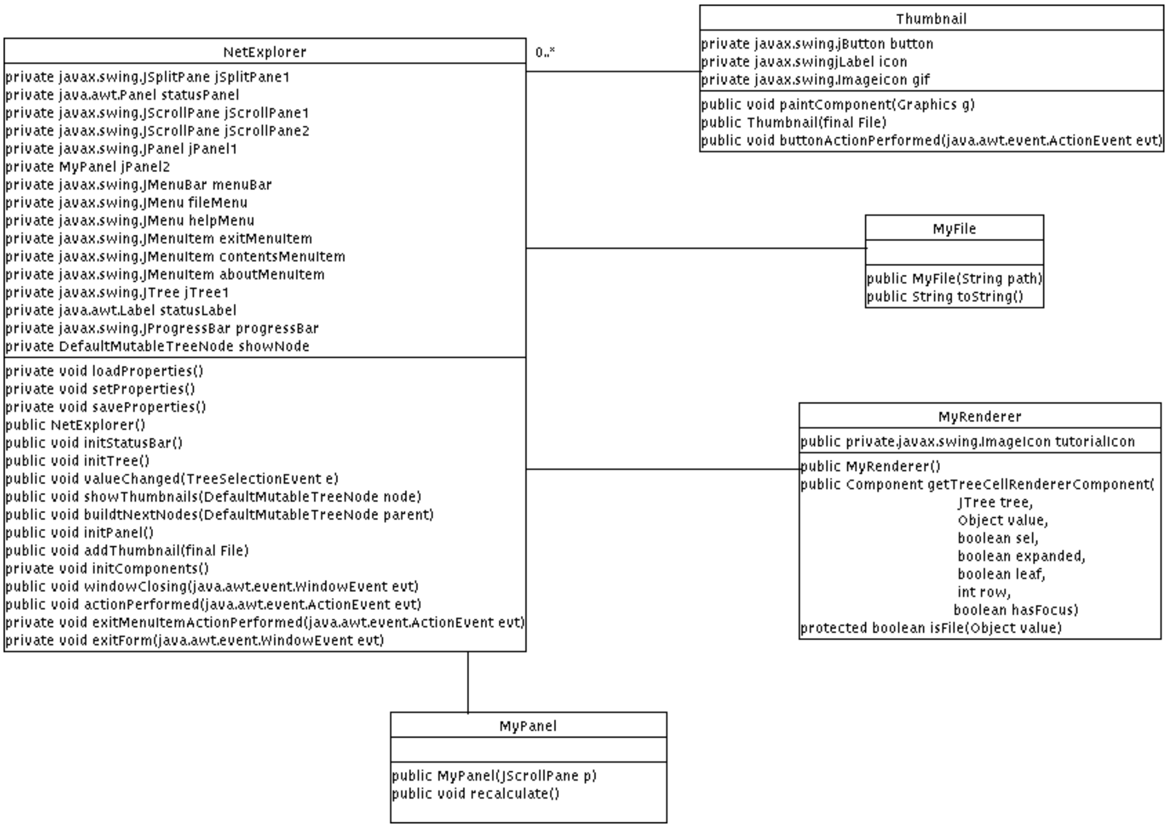

Abbildung 3.1: Class Diagrams von NetExplorer

Kapitel 4

NetViewer

4.1 Beschreibung

Die Java-Applikation NetViewer umfaßt die restlichen Use-Cases. Das Hauptfenster gliedert sich in zwei Teile, im oberen wird die graphische Darstellung des Netzes gezeigt, im unteren werden verschieden Information als Text gezeigt.

Für die Darstellung des Netzes werden Properties verwendet, die für jedes Netz extra angelegt werden. Damit lassen sich Hintergrundfarbe, Knotenfarben, Rasterabstand und -art einstellen. Durch Auswählen des Menüpunktes "Properties" im Popup-Menü des Graphikfensters lassen sich die Einstellungen verändern oder auf den Standardwert zurücksetzen, wobei der Anwender die Möglichkeit hat, selber einen Standardwert zu definieren. Ein einfacher Klick auf einen Knoten ändert den Focus des Knotens. Focusierte Knoten sind an der Umrandung der Datenfelder erkennbar. Da sich auch mehrere Knoten gleichzeitig focusieren lassen, kann man diese Funktion verwenden um die Eigenschaften mehrerer Knoten gleichzeitig zu verändern. Durch Doppelklicken auf einen Knoten öffnet der Anwender ein Popup-Menü, in dem der Anwender mehrere Auswahlmöglichkeiten hat:

- show prop dialog: öffnet das Fnster, in dem der Anwender die Graphischen Eigenschaften eines Knotens verändern kann.
- show details: Öffnet ein Fenster mit detailierten Knotendaten. Diese Funktion ist noch nicht fertig ausgebaut.
- copy properties: kopiert die Eigenschaften dieses Knotens und speichert sie im Properties-Dialog ab.
- paste properties: Ändert die Eigenschaften des Knotens entsprechend der Einstellungen im Properties-Dialog

- paste properties to all focused: Ändert die Eigenschaften aller focusiertend Knoten entsprechend der Einstellungen im Properties-Dialog. Dies Funktion entspricht der Taste "apply" im Properties-Dialog.
- (de)activate: Ändert den Aktivierungs-Zustand des Knotens. Je nach Zustand ändert sich auch die Darstellung des Knotens entsprechend den Einstellungen im Properties-Dialog.

4.2 Implementierte Funktionen

- Netz aus SNNS-File lesen und darstellen.
- Properties der Netzdarstellung ändern und speichern.
- Properties der Units ändern und speichern
- Properties von Units kopieren und einfügen
- Knoten-Daten darstellen abhängig von Zustand.
- Knoten bzw. Datenfelder verschieben
- Knotenzustand ändern
- Speichern von Thumbnails
- Grafik auf *.PS Files exportieren
- Popup-Menü für Knoten:
  - show prop dialog
  - copy properties
  - paste properties
  - activate/deactivat
- Properties-Dialog für Knoten
  - Darstellungsmodus der Datenfelder(immer sichtbar/nie sichtbar/sichtbar wenn aktiviert)
  - Farben der Datenfelder(im aktivierten / nicht aktivierten Zustand)
  - Form der Datenfelder (box/rounded box/circle/no outline)

4.3 Class Diagrams

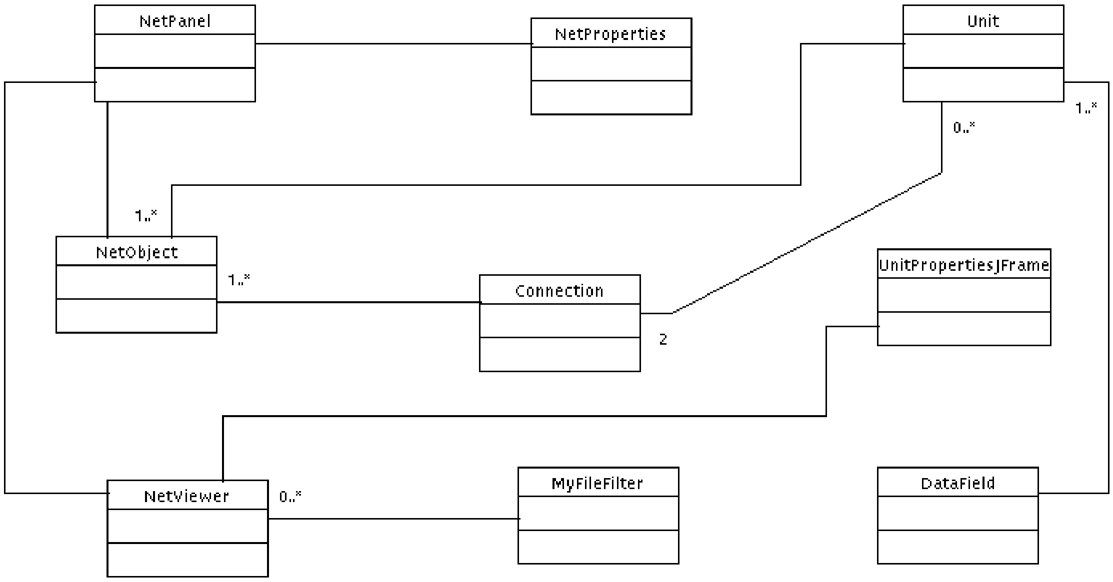

Abbildung 4.1: Class Diagrams von NetViewer

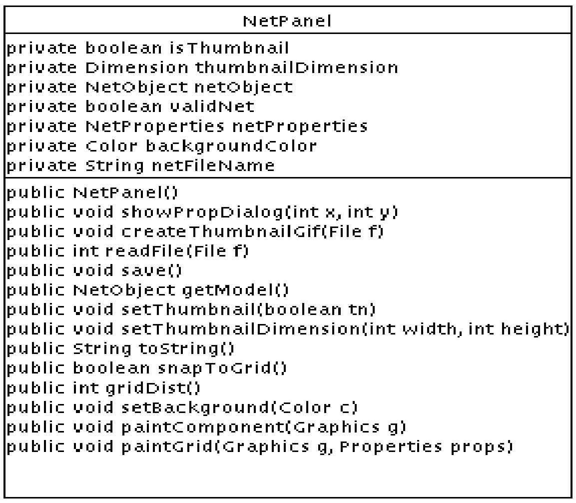

Abbildung 4.2: NetPanel

Abbildung 4.3: NetProperties

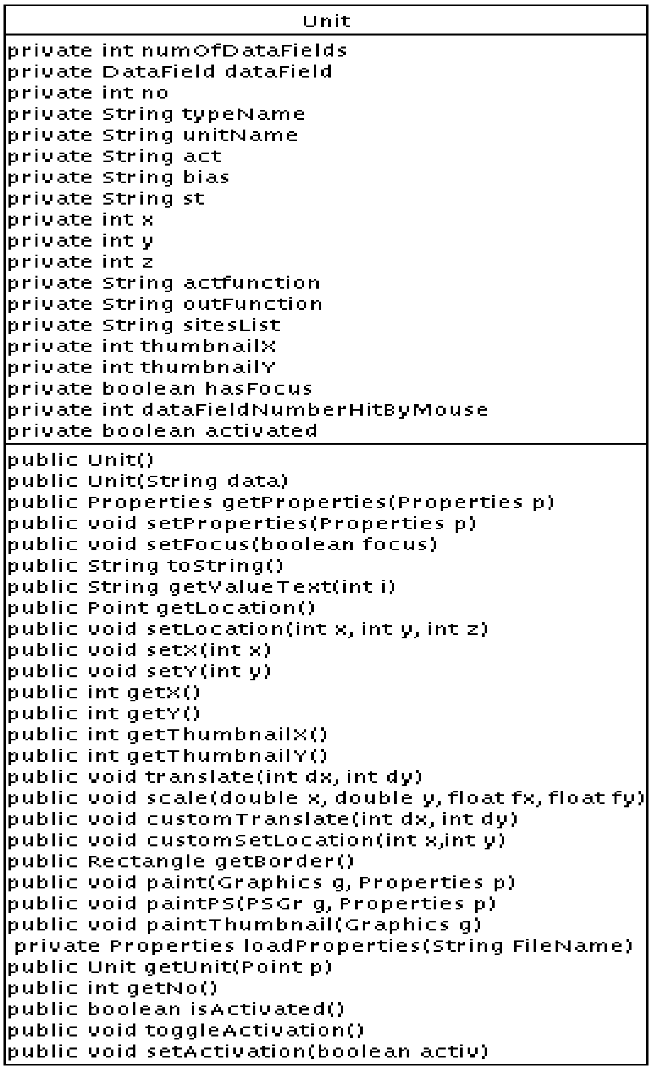

Abbildung 4.4: Unit

NetObject
private int sizeX private int sizeY private String generatedAt private String networkName private String sourceFiles private int noOfUnits private int noOfConnections private String noOfUnitTypes private String noOfSiteTypes private String learningFunction private String pruningFunction private String ffLearningFunction private String updateFunction public String GENERATED_AT public String UNIT_SECTION_TITLE public String CONNECTION_SECTION_TITLE public String NO_OF_UNITS public String NO_OF_CONNECTIONS public int MAX_UNITS private int unitCount public int MAX_CONNECTIONS private int connectionCount
public void save (String netFileName) public void readPropsFile(String netFileName) public int readSNNSFile(File f) public String toString() public void paintPS(PSGr g, Properties p) public void paintComponent(Graphics g, boolean thumbnail, Properties p) public void paintThumbnail(Graphics g) public void setSize(int sizeX, int sizeY) public void scaleTo(int x, int y) public Rectangle getBorder() public void translate(int dx, int dy) public Object getObject(Point p)

Abbildung 4.5: NetObject

Connection
private Unit target private String site private Unit source private String weight
public Connection(Unit target, Unit source, String weight, String site) public void paint(Graphics g, Properties p) public void paintPS(PSGr g, Properties p) public void paintThumbnail(Graphics g) public String toString() private void drawLine(Graphics g, int x1, int y1, int x2, int y2)

Abbildung 4.6: Connection

```
NetViewer
----------------------------------------------------------------
private javax.swing.JSplitPane splitPane
private javax.swing.JScrollPane topScrollPane
private javax.swing.JScrollPane bottomScrollPane
private javax.swing.JTextArea textArea
private javax.swing.JMenuBar menuBar
private javax.swing.JPopupMenu popupMenu
private javax.swing.JMenu fileMenu
private javax.swing.JMenu editMenu
private javax.swing.JMenu helpMenu
private javax.swing.JMenuItem openMenuItem
private javax.swing.JMenuItem saveMenuItem
private javax.swing.JMenuItem saveAsMenuItem
private javax.swing.JMenuItem exitMenuItem
private javax.swing.JMenuItem cutMenuItem
private javax.swing.JMenuItem copyMenuItem
private javax.swing.JMenuItem pasteMenuItem
private javax.swing.JMenuItem deleteMenuItem
private javax.swing.JMenuItem contentsMenuItem
private javax.swing.JMenuItem aboutMenuItem
private javax.swing.JMenuItem PropertiesDrawAreaPopup
public static NetPanel netPane = new NetPanel()
----------------------------------------------------------------
private static boolean startedFromMain()
public JFileChooser fc()
private String PropertiesFileName()
private Properties props()
private Unit lastHitUnit()
private Unit lastFocusUnit()
private Point lastHitPoint()
private UnitPropertiesJFrame unitProperties()
public NetViewer(String PropertiesFileName)
private void initActionListeners()
public void actionPerformed(java.awt.event.ActionEvent evt)
private void setProperties()
private void loadProperties()
private void saveProperties()
private void initnet()
public void mouseDragged(java.awt.event.MouseEvent evt)
public void mouseClicked(java.awt.event.MouseEvent evt)
public void mousePressed(java.awt.event.MouseEvent evt)
public void mouseReleased(java.awt.event.MouseEvent evt)
private void initComponents()
public void windowClosing(java.awt.event.WindowEvent evt)
private void PropertiesDrawAreaPopupActionPerformed(java.awt.event.ActionEvent evt)
private void copyPropertiesUnitPopupActionPerformed(java.awt.event.ActionEvent evt)
private void showPropertiesUnitPopupActionPerformed(java.awt.event.ActionEvent evt)
private void pastePropertiesUnitPopupActionPerformed(java.awt.event.ActionEvent evt)
private void activateUnitPopupActionPerformed(java.awt.event.ActionEvent evt)
private void saveMenuItemActionPerformed(java.awt.event.ActionEvent evt)
private void drawAreaMousePressed(java.awt.event.MouseEvent evt)
private void drawAreaMouseReleased(java.awt.event.MouseEvent evt)
private void drawAreaMouseDragged(java.awt.event.MouseEvent evt)
private void drawAreaMouseClicked(java.awt.event.MouseEvent evt)
public void run()
private void openMenuItemActionPerformed(java.awt.event.ActionEvent evt)
private void cutMenuItemActionPerformed(java.awt.event.ActionEvent evt)
private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt)
private void exitForm(java.awt.event.WindowEvent evt)
private void exportMenuItemActionPerformed(java.awt.event.ActionEvent evt)
private void gifMenuItemActionPerformed(java.awt.event.ActionEvent evt)
```

Abbildung 4.7: NetViewer

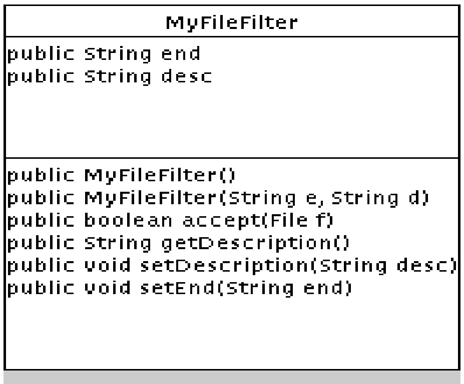

Abbildung 4.8: MyFileFilter

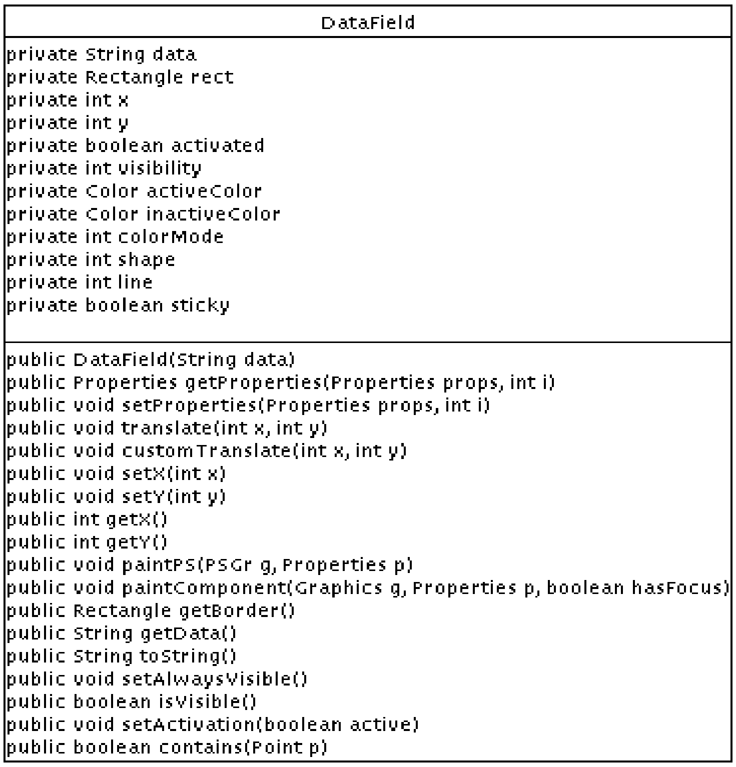

Abbildung 4.9: DataField

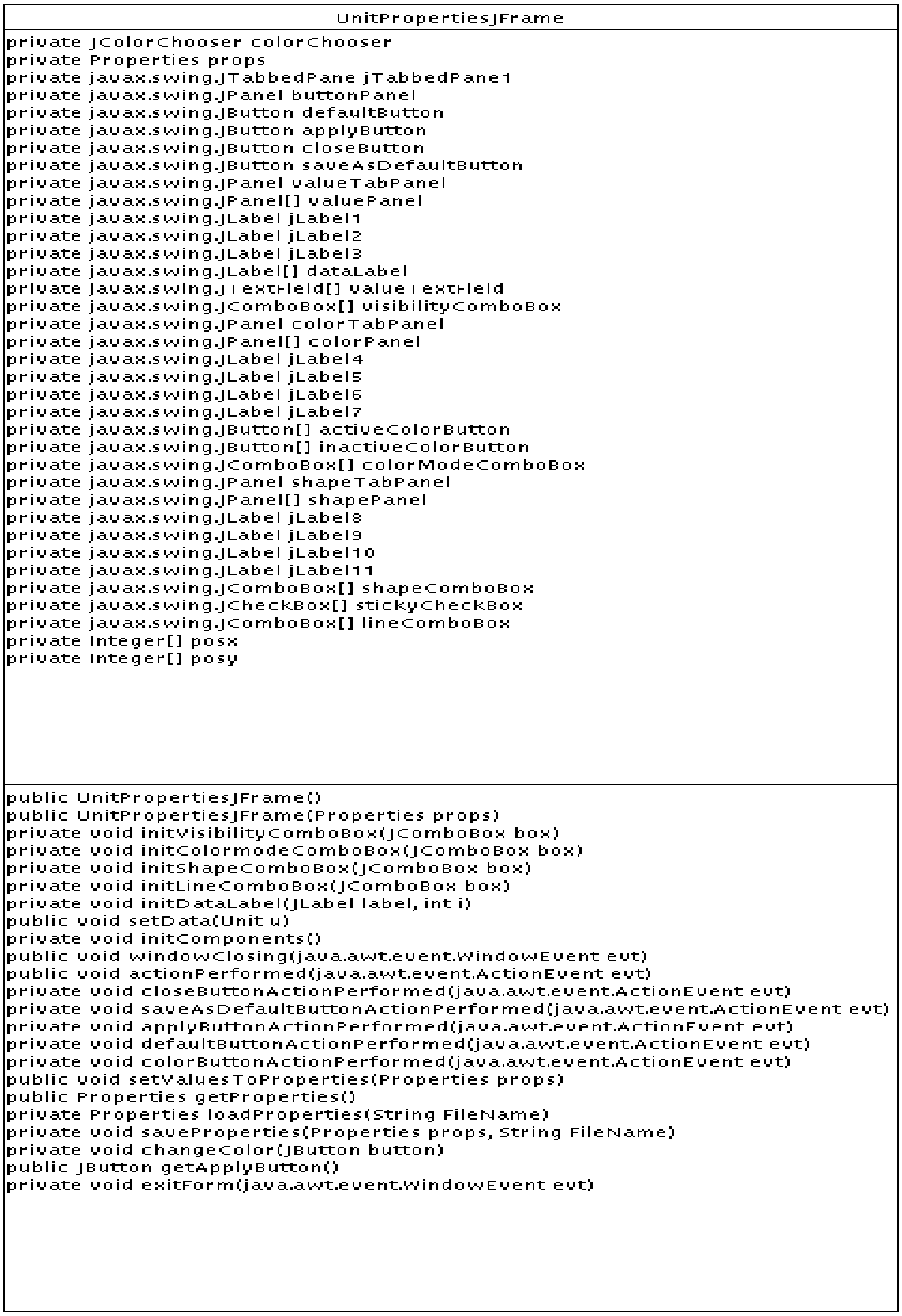

Abbildung 4.10: UnitPropertiesJFrame

Anhang A

Diagramme

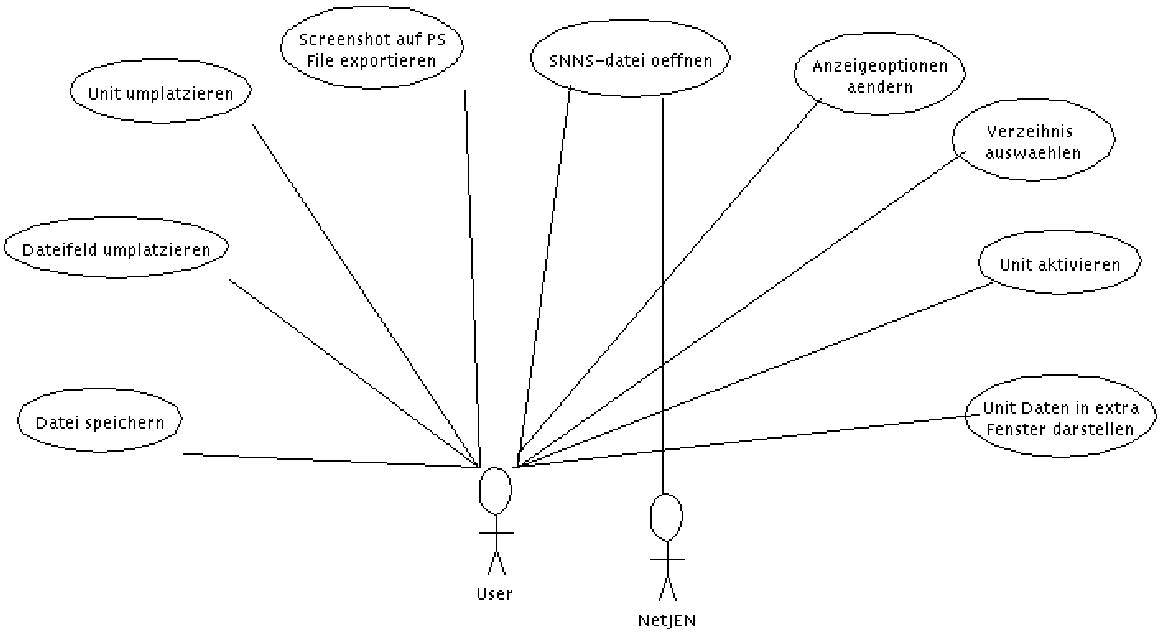

Abbildung A.1: Use-Case Diagramm

Anhang B

Protokolle

B.1 14.3.2000

Teilnehmende Personen: Schwaiger Roland
Lackner Andreas
Jestratijevic Marko

Protokollführer: Andreas Lackner
Ort: Büro Herr Schwaiger
Zeit: 16.30 - 17.00

Einführende Besprechung über das Projekt.
Programmfunktionen:

- Darstellung von ANNs, die aus SNNS-Daten gelesen werden.
- Manipulation der Darstellung am Bildschirm (mit der Maus).
- Drucken und Speichern der Daten in das SNNS-File.

Das Praktikum soll unter Zuhilfenahme folgender Softwarepakete durchgeführt werden:

- UML für den Entwurf und Design
- Java als Programmiersprache
- LaTeXfür die Dokumentation des Projekts

Bis zur Besprechung am 21.3.2000 sollten folgende Daten gesichtet werden:

- Dokumentation UML

- Beschreibung des SNNS Datenformats
- Latex Kurzbeschreibung

B.2 21.3.2000

Teilnehmende Personen: Schwaiger Roland
Lackner Andreas
Jestratijevic Marko

Protokollführer: Andreas Lackner
Ort: Büro Herr Schwaiger
Zeit: 16.30 - 17.00

Grundsätzliche Klärung der Ziele bei der Anwendung von UML

Erste Fragestellungen:

- Welche Daten sind in SNNS-Files gespeichert, was stellen diese Daten dar?
- Wie kann man ANNs graphisch darstellen?
- Welche Netztypen gibt es?

B.3 28.3.2000

Teilnehmende Personen: Schwaiger Roland
Lackner Andreas
Jestratijevic Marko

Protokollführer: Andreas Lackner
Ort: Büro Herr Schwaiger
Zeit: 16.30 - 17.00

Aussehen der Protokolle/Dokumentation.

Diskussion der bisher gefundenen Use Cases. Festlegung darauf, das das Programm in mehrere getrennt voneinander laufende Prozesse aufgegliedert wird. Damit soll auch ein Thread aus dem restlichen NetJEN-Paket unsere Threads aufrufen können (RMI, Remote Method Invocation).

Diskussion des Entwurfs der graphischen Darstellung.

UML-Anwendung bei:

- Use Cases
- sequence diagram
- colaboration diagram

Dazu soll ArgoUML verwendet werden. Ein Beispiel zur Anwendung von UML liegt als Artikel aus c't 1999, Heft 21, vor. ArgoUML ist auf den UNIX-Workstations im Rechnerübungsraum installiert.

Im Rechnerübungsraum liegt eine Beschreibung in BNF der SNNS .net-Files auf.

B.4 11.4.2000

Teilnehmende Personen: Schwaiger Roland
Lackner Andreas
Jestratijevic Marko

Protokollführer: Andreas Lackner
Ort: Büro Herr Schwaiger
Zeit: 16.30 - 17.00

Diskussion der Use Cases.
Diskussion der net.props-Files. Die Zusätzlichen Daten, die zum vollständigen Abspeichern der Graphischen Darstellung benötigt werden, sollen in ein XML Dokument erzeugt werden. Die Unterlagen zur Einführung in den XML-Standard liegen als Kopie vor.
Bis zum nächsten Treffen soll eine Prototyp der GUI vorliegen (JAVA-Programm ohne Funktionalität). Die Klassendiagramme sollen in ArgoUML vorliegen.
Nächster Termin wegen Osterferien und Abwesenheiten erst am 2.5.2000, 16.30 Uhr.

B.5 2.5.2000

Teilnehmende Personen: Schwaiger Roland
Lackner Andreas

Jestratijevic Marko

Protokollführer: Andreas Lackner
Ort: Büro Herr Schwaiger
Zeit: 16.30 - 17.00

Vorstellung der GUI-Prototypen.
Bis zum nächsten Treffen sollen die Prototypen der GUI mit Funktionalität ausgestattet werden.
Nächster Termin am 24.5.2000, 16.30 Uhr.

B.6 24.5.2000

Teilnehmende Personen: Schwaiger Roland
Lackner Andreas
Jestratijevic Marko

Protokollführer: Andreas Lackner
Ort: Workstation-Raum
Zeit: 16.30 - 17.00

Vorstellung und Diskussion der implementierten Funktionalität.
Noch zu implementieren:

- NetExplorer:
  - Icons im Verzeichnisbaum ändern.
  - Verzeichnisbaum aktualisieren, wenn sich Verzeichnisse ändern.
  - Anordnung der Thumbnails verbessern.
  - Startverzeichnis des Baums vom Anwender veränderbar machen.
  - Aufruf des NetViewers als eigenen Thread. Wenn mehr als ein NetViewer gestartet wird, gibt es Probleme.
  - Drag & Drop zwischen NetExplorer und NetViewer
  - Statusbar in der untersten Zeile des Fenster
  - ToolTips bei zu langen Dateinamen
- NetViewer:
  - Properties mit Namen des Netzes abspeichern, damit für jedes Netz verschiedene Properties möglich sind.

- Properties-Dialog für Knoten
  * Darstellungsmodus der Datenfelder (ein/aus/sichtbar wenn aktiviert)
  * Farben der einzelnen Datenfelder (für verschieden Zustände der Unit)
- Popup-Menü für Knoten bzw. Verbindungen:
  * properties
  * activate/deactivat
  * show data
  * copy properties
  * paste properties
- Datenfelder des Knotens verschiebbar.
- Grafik exportieren

Nächster Termin am 7.6.2000, 16.30 Uhr.

Anhang C

Literatur

- Java 1.2 Laura Lemay,Rogers Cadenhead ,pages 649,kompijuter biblioteka Sams publishing
- java homepage (http://www.java.sun.com)
- http://www.acme.com/java/
- SNNS Manual

Anhang D

Erweiterungsvorschläge

- NetViewer
  - Graphische Darstellung
    - Frei einfügbare Textfelder
    - Zoom in/out
    - Automatische Ausrichtung nach Ebenen
    - 3D-Darstellungen
  - Bearbeitung von Netzen
    - Connections bearbeiten
    - Funktionale Werte änderen
    - Units / Connections einfügen
    - Units / Connections löschen
  - Datenaustausch
    - Export auf SNNS-Format
    - Import von SNNS-Dateien als Bestandteil eines bestehenden Netzes
- NetExplorer
  - Graphische Darstellung
    - Veränderbare Thumbnailgröße