



Anwenderdokumentation

Beleg Neuronale Netze

Self-Organizing Maps für Ebenenentfaltung, Tierverwandtschaft,
Handelsreisender und Alphabet

Auswirkungen eines lokal begrenzten neurologischen Netzausfalls auf SOMs.

Dresden, Dezember 2003

Rene Höhne (s4142 - ia00)

Inhaltsverzeichnis

1. Self-Organizing Maps	3
1.1. Grundlagen	3
1.2. Initialisierung	3
1.3. Gewinnerfunktion	3
1.4. Topologische Nachbarschaftsfunktion	4
1.5. Adaption der Gewichte	5
2. SOM-Programm	6
2.1. SOM-Menü	6
2.2. Settings	7
2.3. Init Net	8
2.4. Start	9
3. Beispiele	9
3.1. Ebenenentfaltung (Plane unfolding)	9
3.2. Tierverwandtschaft (Default)	10
3.3. Handelsreisender (Travelling Salesman)	11
3.4. Alphabet (Alphabet)	11

1. Self-Organizing Maps

1.1. Grundlagen

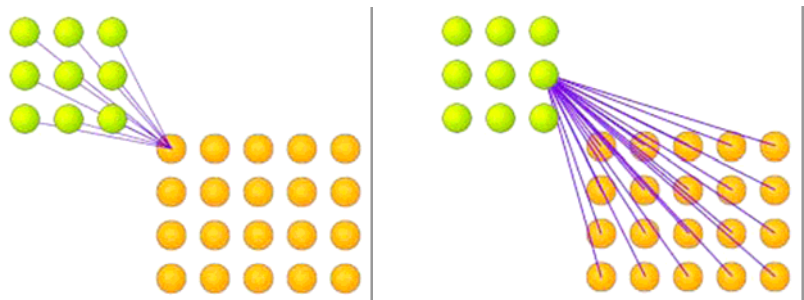
Ziel meines Beleges war es, mit einem Tool die Möglichkeit zu schaffen, die 4 Beispiele zu SOM aus der Vorlesung mit vielen eigenen Parametereinstellungen auszutesten. Des Weiteren sollte noch die Möglichkeit eröffnet werden, ein lokal begrenzten neurologischen Netzausfall einzubauen und auszuwerten.

Dazu verwendete ich selbstorganisierenden Karten (SOM), die in den sechziger Jahren von Professor T. Kohonen entwickelt wurden. Sie werden auch Kohonen-Karten oder -Netzwerke genannt.

Diese Netzwerke bestehen aus mindestens zwei Schichten, der Eingabeschicht und der Ausgabeschicht. Die Ausgabeschicht wird auch Kohonenschicht genannt.

Jedes Neuron der Inputschicht (Pattern) ist mit jedem Neuron der Kohonenschicht verbunden (siehe

Abbildung). Innerhalb der Kohonenschicht sind die jeweils benachbarten Neuronen außerdem noch untereinander gitterartig verbunden. Ich habe eine rechteckige Anordnung gewählt, es sind allerdings auch andere möglich.



3x3 Neuronen großer Inputvektor verbunden mit 5x4 Neuronen großem Kohonenvektor. Übersichtlichkeitshalber sind nur die Verbindungen der Inputneuronen zu einem Kohonen-Neuron (linke Abb.) und die Verbindungen von einem Inputneuron zu allen Outputneuronen dargestellt (rechte Abb.)

Jedes dieser Kohonen-Neuronen hat einen Gewichtsvektor:

$$\vec{w}_j = (w_{1j}, w_{2j}, \dots, w_{mj})^T \quad j = 1, \dots, m$$

Die Neuronen der Inputschicht sind über diesen Gewichtsvektor mit den Neuronen der Kohonenschicht verbunden.

Das Prinzip der SOM besteht nun darin, dass die Neuronen der Kohonenschicht ihre Gewichte so anpassen können, dass charakteristische Inputmerkmale auf einem bestimmten Punkt der Output-Schicht dargestellt werden können.

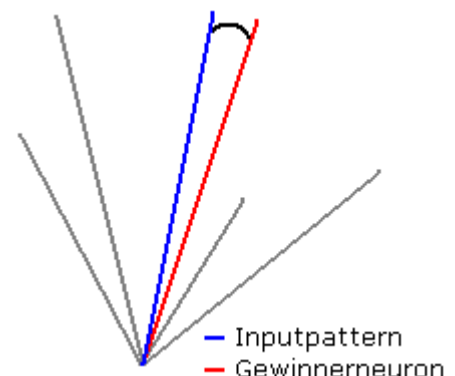
1.2. Initialisierung

Die Gewichte der Outputschicht werden zufällig innerhalb eines festgelegten Intervalls initialisiert.

1.3. Gewinnerfunktion

Aus den Inputpattern wird zufällig ein Inputpattern ermittelt, der an die Gewichtsvektoren jedes Kohonen-Neurons angelegt wird.

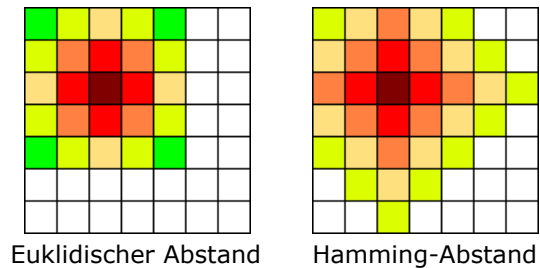
Das Neuron, welches dem Eingabevektor des Inputpattern am ähnlichsten ist (genau dann, wenn der Winkel zwischen Eingabe- und Gewichtsvektor am kleinsten ist), gewinnt den Vergleich und wird somit zu Gewinnerneuron (siehe Abbildung).



1.4. Topologische Nachbarschaftsfunktion

1.4.1. Allgemein

Die Nachbarschaftsfunktion ist ein wichtiger Faktor im Kohonen-Netzwerk. Außer dem Gewinnerneuron werden entsprechend dem vorgegebenen Radius auch noch die umliegenden Neuronen angepasst (siehe Abbildung).



Die Distanz der einzelnen Neuronen zum Eingabevektor des Inputpattern \mathbf{x} wird wie folgt berechnet:

$$d_j = \|\underline{x} - \underline{w}_j\| = \sqrt{\sum_{i=1}^N (x_i - w_{ij})^2}$$

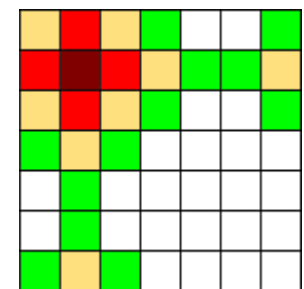
Für die Berechnung der Nachbarschaft können unterschiedliche Funktionen (NBF) zum Einsatz kommen. Es empfiehlt sich eine „weiche“ NBF zu wählen, da dadurch ein sanfterer Übergang zwischen den benachbarten Neuronen gewährleistet wird. Die Berechnung der Distanz (Nachbarschaft) r zwischen Gewinner- und aktuellem Neuron erfolgt mittels Pythagoras (Euklidischer Abstand) oder Summe der Achsendistanzen (Hamming-Abstand).

1.4.2. Nachbarschaftsfunktionen

- Stufenzylinder-Funktion
- Exponentielle Funktion
- Kosinus-Funktion
- Sinus-Funktion
- Tangens-Funktion
- (1 - X) -Funktion

1.4.3. Geschlossenes Netz

Eine Besonderheit stellt das geschlossene Netz dar. Wie in der Abbildung zu erkennen ist, werden die dem Rand gegenüberliegenden Neuronen als Nachbarn angesehen. Somit werden auch die Gewichte über den Rand hinaus adaptiert. Der Effekt ist ein besseres Ausbreiten der Pattern, die sich am Rande etablieren. Dieser Vorteil wirkt sich bei Netzen negativ aus, bei denen die Anzahl der Neuronen gleich der Anzahl der Inputmuster ist.



Geschlossenes Netz mit Hamming-Abstand

1.5. Adaption der Gewichte

Die Gewichtsvektoren der innerhalb des Erregungszentrums des Gewinnerneurons liegenden Kohonen-Neuronen werden mit der Lernrate adaptiert.

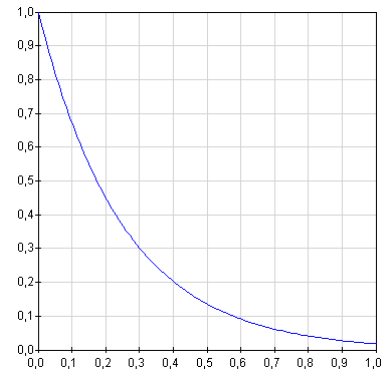
$$w_{ij}^{neu} = w_{ij}^{alt} + \varepsilon \left\| \underline{x} - \underline{w}_j \right\| \text{ falls } j \in N_c$$

1.5.1. Lernrate

Der Wert der topologischen Nachbarschaft geht mit der Lernrate ε in die Adaption der Gewichte ein. Die Lernrate bezeichnet die Stärke der Aktivierung der Synapsengewichte an den Outputneuronen. Sie wird im Laufe des Trainings der SOM stetig verringert.

Dazu stehen im Programm mehrere Funktionen zur Verfügung. Die einfachste Methode ist die lineare Verringerung der Lernrate. Die elegantere Lösung ist die Verringerung mittels einer exponentiellen Funktion. Hier die entsprechende Formel:

$$f(x) = e^{-A \cdot x^B} \quad \varepsilon(x) = \min + (\max - \min) * f(x)$$

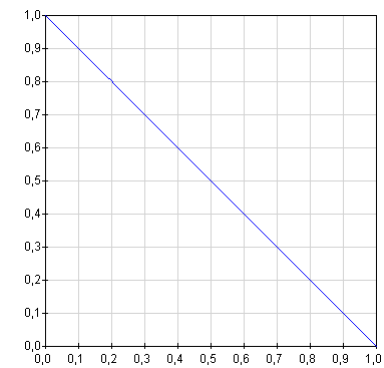


Exponentielle Funktion
(A = 4, B = 1)

1.5.2. Nachbarschaftsradius

Für den zeitlichen Verlauf des Nachbarschaftsradius n gibt es ebenfalls mehrere Möglichkeiten, z.B. die lineare Verringerung oder die wiederum bessere Lösung mittels der exponentiellen Funktion. Sie wird auch im Laufe des Trainings der SOM stetig verringert wie die Lernrate. Hier die Formel für die lineare Verringerung:

$$f(x) = (1 - x) \quad n(x) = \min + (\max - \min) * f(x)$$



Lineare Funktion

1.5.3. topologische Nachbarschaft

Für die topologischen Nachbarschaft gibt es ebenfalls mehrere Variationen, z.B. die exponentielle Nachbarschaft oder die bessere Lösung mittels der linearen Funktion.

1.5.4. Lernphase

Die **Ordnungsphase** ist ausgehend von der Initialisierung der Gewichte die erste Lernphase und durch eine relativ geringe Anzahl von Lernschritten (ca. 100) gekennzeichnet. Während dieser Phase wird der Nachbarschaftsradius ständig verringert (siehe 1.5.2.). Als Startwert sollte ein Wert von etwa dem halben Durchmesser der Kohonenkarte gewählt werden. Bei einer Kartengröße von 30 x 30, wäre der Startradius also 15.

Genau wie der Nachbarschaftsradius sollte auch die Lernschrittweite innerhalb der Ordnungsphase stark monoton von einem Wert $\varepsilon = 1..0,8$ auf $\varepsilon \approx 0,1$ fallen.

Auf die Ordnungsphase folgt die **Konvergenzphase**, die mit bis zu 1000 oder 10.000 Wiederholungen gekennzeichnet ist. Erst durch die abgeschlossene Ordnungsphase ist eine klare Darstellung der charakteristischen Inputmerkmale auf der Kohonen-Schicht möglich. Das Lernen gilt als abgeschlossen, wenn die Gewichtsänderungen $\Delta W < \delta$ sind. δ sollte so gewählt werden, dass eine Gewichtsänderung nicht mehr angezeigt wird. Eine weitere Abbruchbedingung ist die Unterschreitung des Radius von 1.

2. SOM-Programm

2.1. SOM-Menü

2.1.1. Create Net

Nach dem Start von SOM im Menü SOM wählen Sie ‚Create Net‘ und Sie können ein neues Kohonen-Netz erstellen. Hier legen Sie nun die Zeilen, Spalten sowie Inputanzahl des Netzes fest. Sie können jederzeit ihr Kohonen-Netz verändern, die Einstellungen und das Pattern bleiben Ihnen erhalten.

Weitere Festlegungen zum Kohonen-Netz können Sie im Menüpunkt ‚Settings‘ treffen (siehe 2.2.).

2.1.2. Create/ Edit Pattern

Im Menü SOM wählen Sie ‚Create/Edit Pattern‘. Nun können Sie ein neues Pattern erstellen oder ein geladenes bearbeiten.

Zum Erstellen legen Sie den Namen und den Typ des Patterns fest.

Bei einem nicht ‚Default‘-Pattern-Typ erscheint eine Erstellungshilfe, die Sie benutzen können, um schnell Patterns zu erstellen.

Geben Sie jedem Pattern-Eintrag einen eindeutigen und möglichst kurzen Namen.

Mit den Pfeilen ‚<<‘, ‚<‘, ‚>‘ und ‚>>‘ können Sie durch alle Pattern-Einträge navigieren.

Sie können auch eigene Patterns erstellen. Nutzen Sie dazu im ‚Default‘

und ‚Displaying Number‘-Typ die Eingabefelder. Im ‚Alphabet‘-Typ die weißen Felder (Klick auf ein Feld, wechselt die Farbe zwischen weiß und schwarz.) und im Travelling-Typ kann man die Eingabefelder unter den Pfeilen nutzen oder den Rechts-Klick der Maus im Diagramm.

2.1.3. Load Pattern-File

Dieser Menüpunkt dient zum Öffnen eines abgespeicherten Patterns.

In der „pat“-Datei sind nur die Patterns mit Namen, Typ und Eingabevektor im XML-Format (SOAP) gespeichert.

2.1.4. Save Pattern to File

Dieser Menüpunkt dient zum Speichern eines Patterns.

2.1.5. Load SOM-File

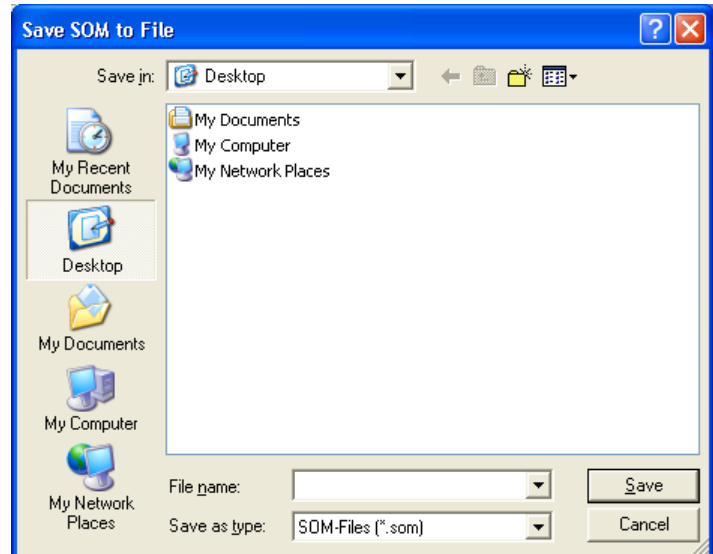
Dieser Menüpunkt dient zum Öffnen eines abgespeicherten SOMs (Projekt).

2.1.6. Save SOM to File

Hiermit sichern Sie ein komplettes SOM - Projekt.

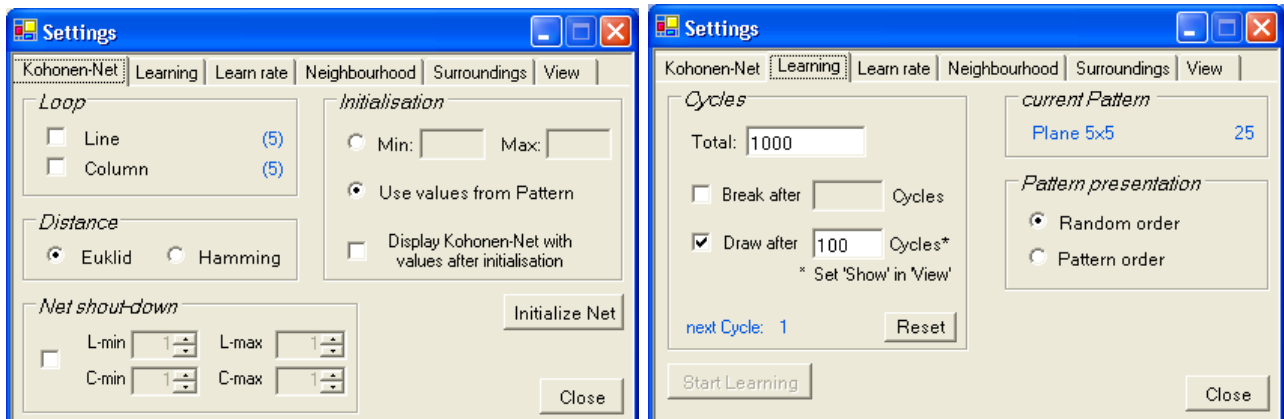
In der Datei ist sowohl das Kohonen-Netz mit allen aktuellen Werten, das Pattern sowie alle Einstellungen (Settings) gespeichert.

SOM - Projekte werden mit der Endung *.som im XML-Format (SOAP) gespeichert.



2.2. Settings

2.2.1. Kohonen Net und Learning



Kohonen Net

Hier kann man viele Eigenschaften des Netzes einstellen.

Loop

Zeilenanfang und -ende verbinden (als Ring) – analog bei der Spalte

Distance

Euklidischer oder Hamming-Abstand

Net shout-down

Gebiet in dem die Neuronen inaktiv/ nicht ansprechbar sind.

Initialisierung

Mit welchen Werten wird das Kohonen Netz initialisiert (eigene oder aus Pattern).

Initialize Net

Initialisiere das Netz in der gewählten Art.

Learning

Hier kann man Einstellungen zum Lernen vornehmen.

Cycles

Total ... Gesamtanzahl der Durchläufe (alle Pattern einmalig angelegt)

Break ... Unterbreche das Lernen alle **x** Durchläufe

Draw ... Zeichne den Graphen alle x Durchläufe neu.

Dies unterbricht nicht das Lernen.

Reset ... Setzt den Counter "next Cycle" zurück.

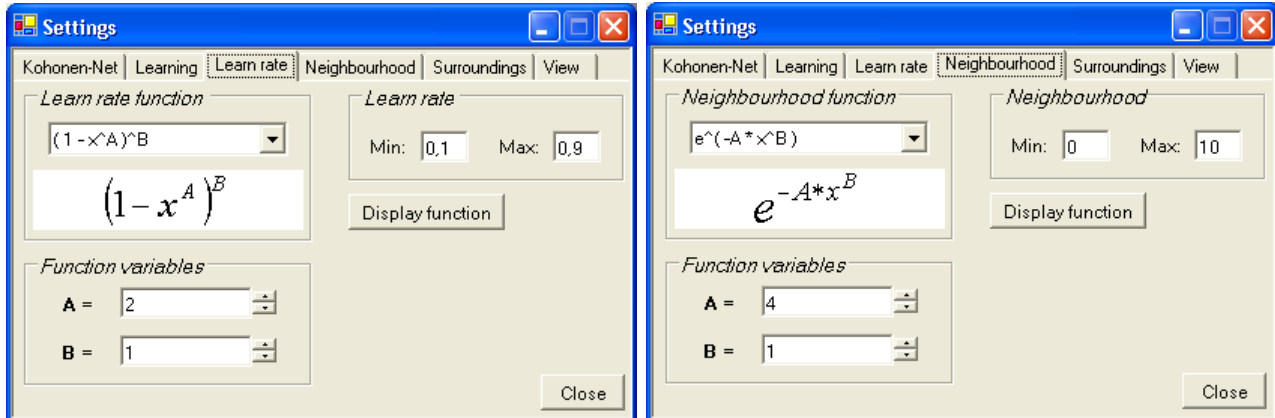
Pattern presentation

Die Pattern werden dem Netz in zufälliger oder Pattern-Reihenfolge präsentiert.

Start Learning

Startet das Lernen des Netzes.

2.2.2. Learn rate und Neighbourhood



Learn rate & Neighbourhood

Hier kann man die Lernrate und Nachbarschaft einstellen.

Hierzu stehen 6 Funktionen zur Verfügung.

Function

- Stufenzylinder-Funktion
- Kosinus-Funktion
- Tangens-Funktion
- Exponentielle Funktion
- Sinus-Funktion
- (1 - X) -Funktion

Variables

Variablen, mit denen man die Funktion noch Feinjustieren kann.

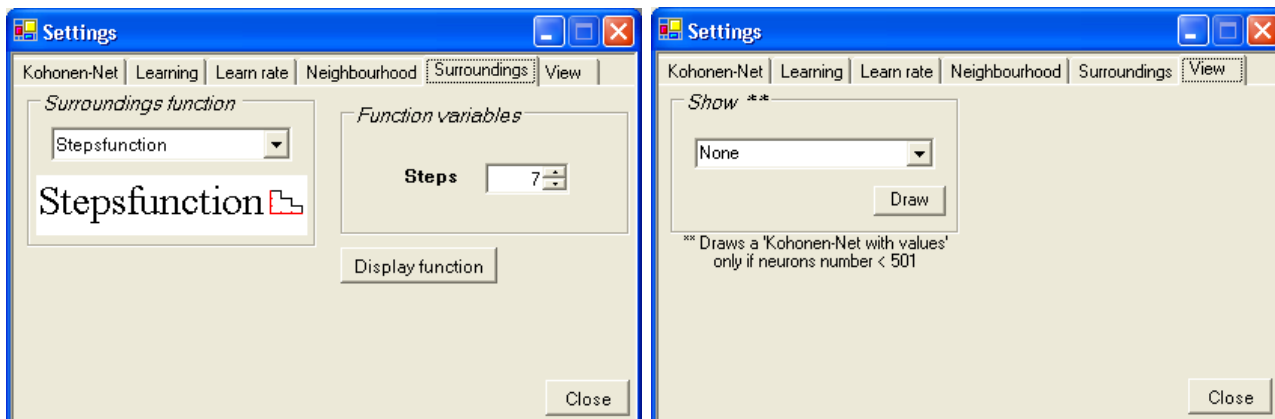
Display function

Mit "Display function" kann man sich die Funktion ansehen.

Learn rate oder Neighbourhood

Hier legt man die max. Lernrate (Startwert) und min. Lernrate (Endwert) fest.

2.2.3. Surrounding und View



Surroundings

Hier kann man die Umgebung einstellen. Hierzu stehen 7 Funktionen zur Verfügung.

Function

- die selben Funktionen wie bei Lernrate aufgeführt
- Mexican Hat-Funktion (nur ein Annäherung)

Variables

Variablen, mit denen man die Funktion noch Feinjustieren kann.

Display function

Mit "Display function" kann man sich die Funktion ansehen.

View

Hier kann man die grafisch Darstellung bestimmen, mit der man das Netz betrachten möchte. Hierzu stehen verschiedene Darstellungen in Abhängigkeit des Patterns zur Verfügung.

Show

- Kohonen Net with values
- Kohonen Net with pattern
- Graph of Champions
- Graph of net unfolding (nur bei Ebenenentfaltung)
- Travelling Salesman Graph (nur bei Handelsreisender)

Mit dem "Draw"-Button kann man auch sofort das Netz betrachten.

2.3. Init Net

Initialisiere das Netz in der gewählten Art (Settings).

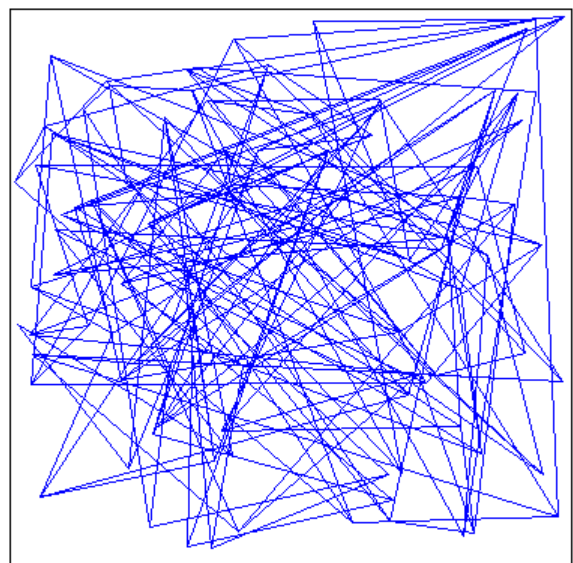
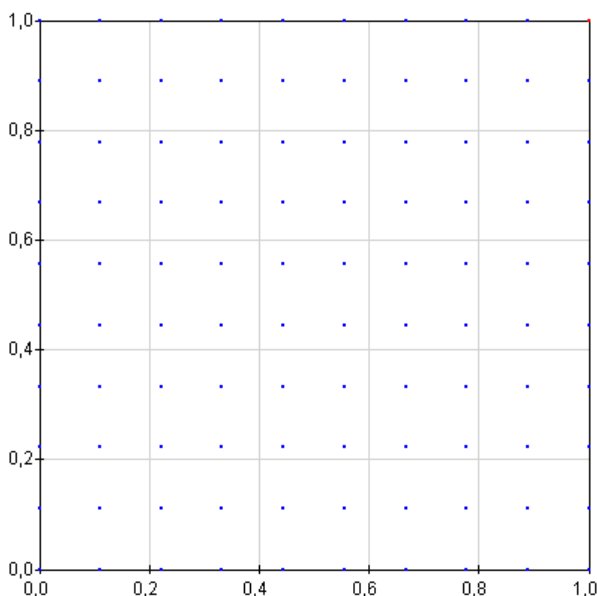
2.4. Start

Startet das Lernen des Netzes.

3. Beispiele

3.1. Ebenenentfaltung (Plane unfolding)

Bei diesem Beispiel wurde ein 10x10x2 Kohonen-Netz verwendet. Mit dem Patterneditor werden automatisch alle 100 Pattern erstellt. Nun nutzt man die Standardeinstellungen des 'Settings' bis auf die Nachbarschaft, die man auf 1-5 festlegt. In den Bildern sieht man die Pattern und die Ebene nach der Initialisierung.

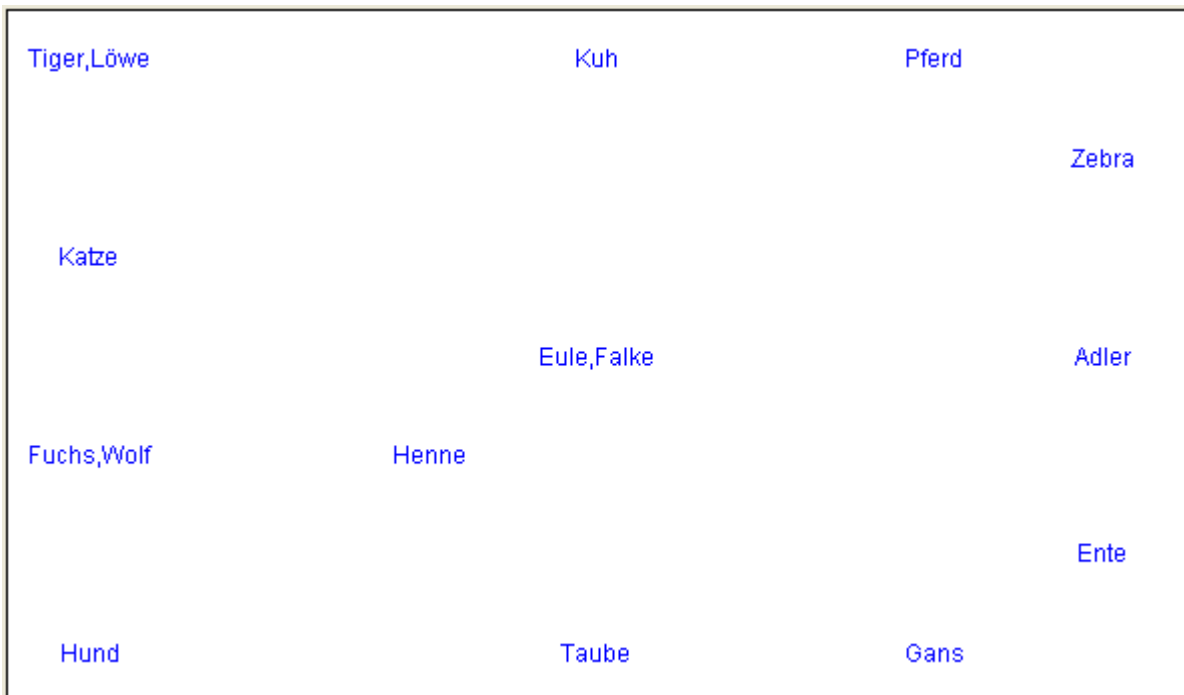


3.2. Tierverwandtschaft (Default)

Bei diesem Beispiel wurde ein 7x7x12 Kohonen-Netz verwendet. Man ladet das Tier-Pattern (16TierePattern_12.pat). Mit dem Patterneditor kann man sich die Werte der 16 Tiere betrachten. Nun nutzt man die Standardeinstellungen des ‚Settings‘ bis auf die Nachbarschaft, die man auf 1-4 festlegt.

Tier	klein	mittel	groß	2 Beine	4 Beine	Fell	Federn	Hufe	jagen	rennen	fliegen	schwim
Taube	1			1			1				1	
Henne	1			1			1					
Ente	1			1			1					1
Gans	1			1			1				1	1
Eule	1			1			1		1		1	
Falke	1			1			1		1		1	
Adler		1		1			1		1		1	
Fuchs		1			1	1			1	1		
Hund		1			1	1				1		
Wolf		1			1	1			1	1		
Katze	1				1	1			1			
Tiger			1		1	1			1	1		
Löwe			1		1	1			1	1		
Pferd			1		1	1		1		1		
Zebra		0,2	0,8		1	1		1		1		
Kuh			1		1	1		1				

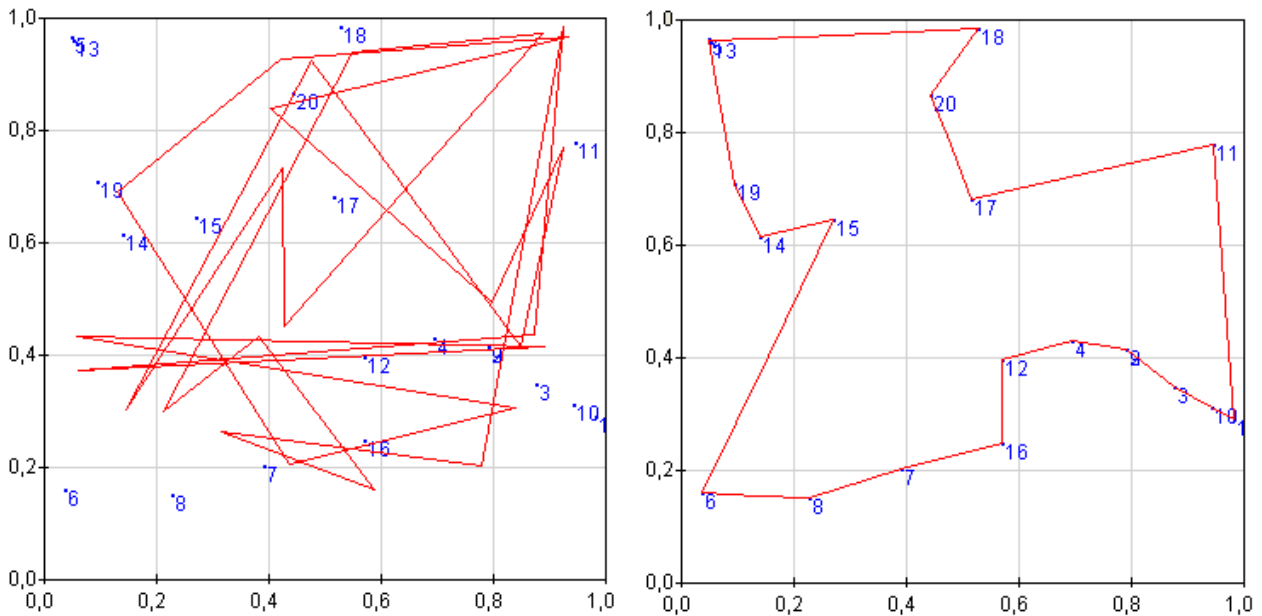
In dem Bild sieht man ein Ergebnisse nach dem Lernen.



3.3. Handelsreisender (Travelling Salesman)

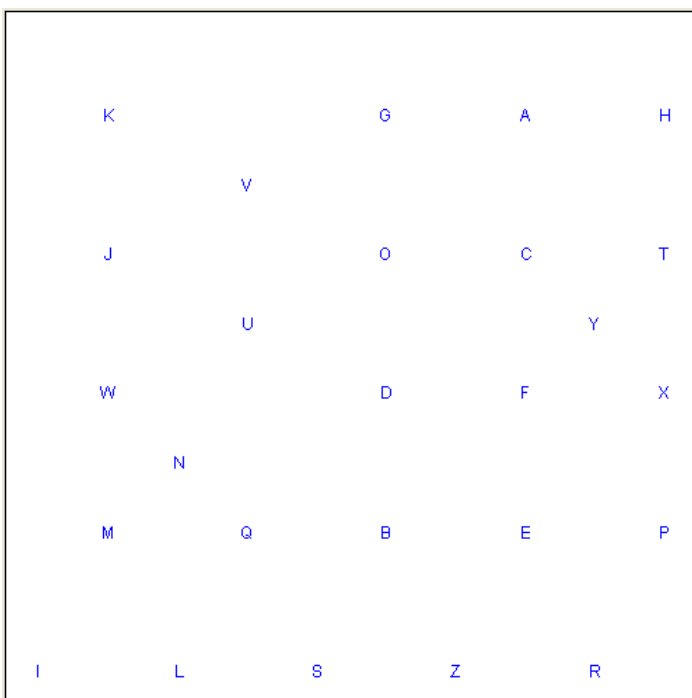
Bei diesem Beispiel wurde ein $1 \times 25 \times 2$ Kohonen-Netz verwendet. Mit dem Patterneditor erstellt man sich (automatisch, oder nicht) 20 Pattern. Nun nutzt man die Standardeinstellungen des ‚Settings‘ bis auf die Nachbarschaft, die man auf 1-15 festlegt und den Zeilen-Loop, den man aktiviert.

Die Bilder zeigen die Pattern und das Netz nach der Initialisierung und dem Lernen.



3.4. Alphabet (Alphabet)

Bei diesem Beispiel wurde ein $10 \times 10 \times 30$ Kohonen-Netz verwendet. Man ladet das Alphabet-Pattern (Alphabet_AZ_30.pat). Mit dem Patterneditor kann man sich die Pattern der Buchstaben betrachten. Nun nutzt man die Standardeinstellungen des ‚Settings‘ bis auf die Nachbarschaft, die man auf 1-5 festlegt und den Zeilen- und Spalten-Loop, die man aktiviert. Hier sieht man das Ergebnis nach dem Lernen.



Bei diesem Beispiel wurde nun noch ein Net shout-down eingefügt.

D.h. ein Gebiet steht nicht mehr zum Lernen zur Verfügung (wie bei einem Schlaganfall).

Verwendet wurden die vorherigen Einstellungen, nur ein Net shout-down von 3 bis 6 für Zeile und Spalte wurde hinzugefügt.

Hier sieht man das Ergebnis nach dem Lernen.

