Selbstorganisierende Karte

Als **Selbstorganisierende Karten**, **Kohonenkarten** oder **Kohonennetze** (nach Teuvo Kohonen; englisch *self-organizing map*, *SOM* bzw. *self-organizing feature map*, *SOFM*) bezeichnet man eine Art von künstlichen neuronalen Netzen. Sie sind als unüberwachtes Lernverfahren ein leistungsfähiges Werkzeug des Data-Mining. Ihr Funktionsprinzip beruht auf der biologischen Erkenntnis, dass viele Strukturen im Gehirn eine lineare oder planare Topologie aufweisen. Die Signale des Eingangsraums, z. B. visuelle Reize, sind jedoch multidimensional.

Es stellt sich also die Frage, wie diese multidimensionalen Eindrücke durch planare Strukturen verarbeitet werden. Biologische Untersuchungen zeigen, dass die Eingangssignale so abgebildet werden, dass ähnliche Reize nahe beieinander liegen. Der Phasenraum der angelegten Reize wird also kartiert.

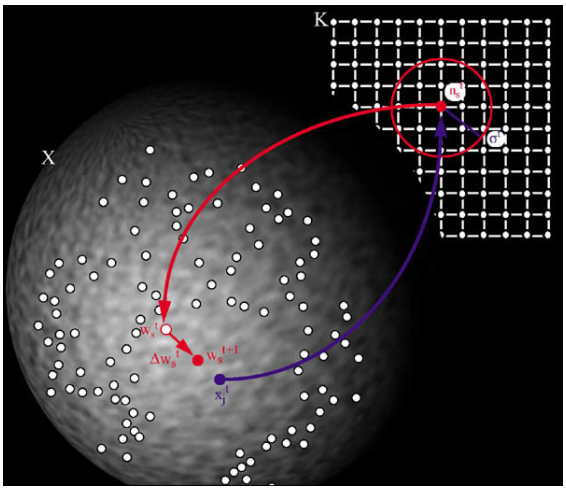
Wird nun ein Signal an diese Karte herangeführt, so werden nur diejenigen Gebiete der Karte erregt, die dem Signal ähnlich sind. Die Neuronenschicht wirkt als *topologische Merkmalskarte*, wenn die Lage der am stärksten erregten Neuronen in gesetzmäßiger und stetiger Weise mit wichtigen Signalmerkmalen korreliert ist.

Anwendung finden selbstorganisierende Karten zum Beispiel in der Computergrafik als Quantisierungsalgorithmus zur Farbreduktion von Rastergrafikdaten und in der Bioinformatik zur Clusteranalyse.

1 Laterale Umfeldhemmung

Ein allgemeines Arbeitsprinzip des Nervensystems ist, dass aktive lokale Gruppen von Nervenzellen andere Gruppen ihrer Umgebung hemmen, und somit deren Aktivität unterdrücken (siehe laterale Hemmung). Die Aktivität einer Nervenzelle wird daher aus der Überlagerung des erregenden Eingangssignals und den hemmenden Beiträgen aller Schichtneuronen bestimmt. Da diese laterale Hemmung überall gilt, kommt es zu einem ständigen Wettbewerb um die Vorherrschaft. Der Verlauf der lateralen Hemmung ist für kurze Distanzen erregend/verstärkend und für lange Distanzen hemmend/schwächend. Es lässt sich zeigen, dass dieser Effekt ausreichend ist, eine Lokalisierung der Erregungsantwort in der Nähe der maximalen äußeren Erregung zu bewirken.

2 Struktur und Lernen



Ein Adaptionsschritt: Der Reiz v zieht an dem Gewichtsvektor w des am besten angepassten Neurons. Dieser Zug wird mit zunehmendem Abstand, gemessen im Competitive Layer vom besten Neuron, zunehmend schwächer. Einfach ausgedrückt, beult sich die Karte in Richtung des Reizes v aus.

Eine Eingabeschicht mit n Neuronen ist vollständig mit allen Neuronen innerhalb der Kohonenkarte (der sogenannte *competitive layer*), im Folgenden einfach Karte, verbunden. Jeder zu kartierende Eingangsreiz v wird über die Verbindungen an jedes Neuron dieser Karte weitergegeben.

Die Verbindungsgewichte w zwischen den Neuronen der Eingabeschicht und den Neuronen in der Karte definieren je einen Punkt im Eingangsraum der angelegten Reize v. Alle Neuronen innerhalb der Karte sind untereinander inhibitorisch (hemmend) vernetzt.

1. Die Abbildung zeigt einen Adaptionsschritt im Modell von Kohonen. Ein Reiz v wird an das Netz angelegt.
2. Das Netz sucht das Erregungszentrum s in der Karte, dessen Gewichtsvektor w am nächsten zu v liegt (kleinster Abstand).
3. Der Unterschied zwischen w und v wird in einem Adaptionsschritt verringert.
4. Die Neuronen nahe am Erregungszentrum s werden auch adaptiert, aber umso weniger, je weiter sie vom Erregungszentrum entfernt sind.

Es ist gebräuchlich, aber nicht zwingend, sowohl für die Lernvektoren als auch für die Karte den euklidischen Abstand als Abstandsmaß zu verwenden.

Steht ein Satz verschiedener Trainingsdaten zur Verfügung, so ist eine *Epoche* im Training vollständig, wenn alle Reize genau einmal in zufälliger Reihenfolge an die Eingabeschicht angelegt worden sind. Das Training endet, wenn das Netz seinen stabilen Endzustand erreicht hat.

Das Lernen in einer selbstorganisierten Karte kann formal als iterativer Prozess beschrieben werden. Im Anfangszustand sind die Gewichtsvektoren der Neuronen zufällig im Netz verteilt und in jedem Lernschritt wird an das Netz ein Reiz angelegt. Die selbstorganisierende Karte verändert die Gewichtsvektoren der Neuronen entsprechend der Hebbschen Lernregel, sodass sich im Laufe der Zeit eine topografische Abbildung ergibt.

3 Training einer SOM im Beispiel

Die folgende Tabelle zeigt ein Netz, dessen Neuronen in einem Gitter angeordnet sind und zu Beginn zufällig im Raum verteilt sind. Es wird mit Eingabereizen aus dem Quadrat trainiert, die gleichverteilt sind.

4 Formale Beschreibung des Trainings

Gegeben ist eine endliche Menge **M** von Trainingsstimuli *mi*, die durch einen n-dimensionalen Vektor *xi* spezifiziert sind:

$$M = \{m_i = (x_i) \mid x_i \in X \subseteq \mathbb{R}^n, i = 1, ..., \mu_M\}$$

Weiterhin sei eine Menge von μN Neuronen gegeben, denen jeweils ein Gewichtsvektor w_i in *X* und eine Position k_i auf einer Kohonen-Karte zugeordnet wird, die im weiteren als zweidimensional angenommen wird. Die Kartendimension kann beliebig-dimensional gewählt werden, wobei Kartendimensionen kleiner-gleich drei zur Visualisierung von hochdimensionalen Zusammenhängen verwendet werden. Die Positionen auf der Karte sollen diskreten, quadratischen Gitterpunkten entsprechen (alternative Nachbarschaftstopologien wie z. B. hexagonale Topologien sind ebenfalls möglich), und jeder Gitterpunkt soll durch genau ein Neuron besetzt sein:

$$N = \{n_i = (w_i, k_i) \mid w_i \in X \subseteq \mathbb{R}^n, k_i \in K^2, i = 1, ..., \mu_N\}$$

In der **Lernphase** wird aus der Menge der Stimuli zum Präsentationszeitpunkt *t* ein Element *mj*^{t} gleichverteilt zufällig ausgewählt. Dieser Stimulus legt auf der Karte ein Gewinnerneuron *ns*^{t} fest, das als **Erregungszentrum** bezeichnet wird. Es handelt sich dabei um genau das Neuron, dessen Gewichtsvektor *ws*^{t} den geringsten Abstand im Raum **X** zu dem Stimulusvektor *xj*^{t} besitzt, wobei eine Metrik *dX*(.,.) des Inputraumes gegeben sei:

$$d_X(x_j^t, w_s^t) = min\{d_X(x_j^t, w_i^t) \mid i = 1, ..., \mu_N\}$$

Nachdem *ns*^{t} ermittelt wurde, werden alle Neuronen *ni*^{t} bestimmt, die neben dem Erregungszentrum ihre Gewichtsvektoren anpassen dürfen. Es handelt sich dabei um die Neuronen, deren Entfernung *dA*(*ks*, *ki*) auf der Karte nicht größer ist als ein zeitabhängiger Schwellenwert, der als **Entfernungsreichweite** δ^t bezeichnet wird, wobei eine Metrik *dA*(.,.) der Karte gegeben sei. Diese Neuronen werden in einer Teilmenge $\mathbf{N^{+t}} \subset \mathbf{N^t}$ zusammengefasst:

$$N^{+t} = \{n_i = (w_i, k_i) \mid d_A(k_s, k_i) \leq \delta^t\}$$

Im folgenden Adaptionsschritt wird auf alle Neuronen aus $\mathbf{N^{+t}}$ ein Lernschritt angewendet, der die Gewichtsvektoren verändert. Der Lernschritt ist interpretierbar als eine Verschiebung der Gewichtsvektoren in Richtung des Stimulusvektors *xj*^{t}.

Es wird entsprechend dem Modell von Ritter et al. (1991) dabei die folgende Adaptionsregel verwendet:

$$w_s^{t+1} = w_s^t + \epsilon^t \cdot h_{si}^t \cdot (x_j - w_s^t)$$

mit den zeitabhängigen Parametergleichungen ε^t und *hsi*^{t}, die festgelegt werden als:

1) Die zeitabhängige Lernrate ε^t:

$$\epsilon^t = \epsilon_{\text{start}} \cdot \left(\frac{\epsilon_{\text{end}}}{\epsilon_{\text{start}}}\right)^{\frac{t}{t_{\max}}}$$

mit der Startlernrate ε_{start} und ε_{end} als der Lernrate zum Ende des Verfahrens, d.h. nach t_{max} Stimuluspräsentationen.

2) Die zeitabhängige Entfernungsgewichtungsfunktion *hsi*^{t}:

$$h_{si}^t = e^{\frac{-d_A(k_s, k_i)^2}{2 \cdot (\delta^t)^2}}$$

mit δ^t als dem Nachbarschafts- oder Adaptionsradius um das Gewinner-Neuron auf der Karte:

$$\delta^t = \delta_{\text{start}} \cdot \left(\frac{\delta_{\text{end}}}{\delta_{\text{start}}}\right)^{\frac{t}{t_{\max}}}$$

mit dem Adaptionsradius δ_{start} zum Anfang des Verfahrens, und δ_{end} als dem Adaptionsradius zum Ende des Verfahrens.

Damit eine topologie-erhaltende Abbildung entsteht, d.h. dass benachbarte Punkte im Inputraum **X** auf benachbarte Punkte auf der Karte abgebildet werden, müssen zwei Faktoren berücksichtigt werden:

1. Die topologische Nachbarschaft hsi^t um das Erregungszentrum muss anfangs groß gewählt und im Laufe des Verfahrens verkleinert werden.

2. Die Adaptionsstärke ε^t muss ausgehend von einem großen Wert im Laufe des Verfahrens auf einen kleinen Restwert sinken.

In dem dargestellten Lernprozess werden t_{max} Präsentationen durchgeführt, wonach die SOM in die **Anwendungsphase** überführt werden kann, in der Stimuli präsentiert werden, die in der Lernmenge nicht vorkamen. Ein solcher Stimulus wird dem Gewinnerneuron zugeordnet, dessen Gewichtsvektor die geringste Distanz von dem Stimulusvektor besitzt, sodass dem Stimulus über den Umweg des Gewichtsvektors ein Neuron und eine Position auf der Neuronenkarte zugeordnet werden kann. Auf diese Weise wird der neue Stimulus automatisch klassifiziert und visualisiert.

5 Varianten der SOM

Es wurden eine Vielzahl von Varianten und Erweiterungen zu dem ursprünglichen Modell von Kohonen entwickelt, u.a.:

- Kontext-SOM (K-SOM)
- Temporäre SOM (T-SOM)
- Motorische SOM (M-SOM)
- Neuronen-Gas (NG-SOM)
- Wachsende Zellstrukturen (GCS-SOM)
- Wachsende Gitterstruktur (GG-SOM)
- Wachsende hierarchische SOM (GH-SOM)
- Wachsendes Neuronen-Gas (GNG-SOM)
- Parametrische SOM (P-SOM)
- Hyperbolische SOM (H-SOM)
- Interpolierende SOM (I-SOM)
- Local-Weighted-Regression-SOM (LWR-SOM)
- Selektive-Aufmerksamkeits-SOM (SA-SOM)
- Gelernte Erwartungen in GNG-SOMs (LE-GNG-SOM)
- Fuzzy-SOM (F-SOM)
- Adaptive-Subraum-SOM (AS-SOM)
- Generative Topographische Karte (GTM)

6 Literatur

- Günter Bachelier: *Einführung in selbstorganisierende Karten*. Tectum-Verlag, Marburg 1998, ISBN 3-8288-5017-0
- Teuvo Kohonen: *Self-Organizing Maps*. Springer-Verlag, Berlin 1995, ISBN 3-540-58600-8
- Helge Ritter, Thomas Martinetz, Klaus Schulten: *Neuronale Netze. Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke*. Addison-Wesley, Bonn 1991, ISBN 3-89319-131-3

7 Weblinks

- ANNetGPGPU: C++ Library mit einer Implementierung für SOMs auf GPUs und CPUs und Python Interface

Commons: Selbstorganisierende Karte – Sammlung von Bildern, Videos und Audiodateien

- SOM-Research an der Helsinki University of Technology (Teuvo Kohonen)
- Über SOM in der comp.ai.neural-nets FAQ
- Java SOMToolbox: Open Source Anwendung zum Erstellen, Analysieren und Interagieren mit Selbstorganisierenden Karten, entwickelt an der Technischen Universität Wien.
- Datenbionik: Datenvisualisierung und Data-Mining mit Emergenten SOM: Prof. Ultsch Marburg
- MusicMiner: Visualisierung von Musiksammlungen ESOM
- GNOD, *The **G**lobal **N**etwork **o**f **D**reams*, ein Kohonen-Netz zur Bestimmung von Ähnlichkeiten von Musik, Film und Buchautoren .
- Demonstrationsbeispiel: HTW Dresden - ein SOM fängt einen Ball
- Viscovery SOMine: SOM Technologie Tool von Viscovery
- Neural Networks with Java

8 Text- und Bildquellen, Autoren und Lizenzen

8.1 Text

- **Selbstorganisierende Karte** *Quelle:* https://de.wikipedia.org/wiki/Selbstorganisierende_Karte?oldid=154247773 *Autoren:* JakobVoss, Zeno Gantner, ErikDunsing, Haegar, Crux, D, Stern, MartinWoelker, Apocalexiz, Nina, LightWolf, Wikisearcher, Doc Taxon, Tobias Denninger, Ahans, Aussendorf, ChristophDemmer, Darian, Botteler, Geisslr, Ixitixel, Formativ, Fossa, Schlurcher, Slow Phil, Opethmetal, Sypholux, STBR, Drahreg01, Chrislb, Tillmann Lübker, Hans Koberger, RobotQuistnix, Hobster~dewiki, JCS, Eskimbot, Semperor, Biengo, Thijs!bot, Bernard Ladenthin, Fleshgrinder, Lars Winterfeld, Tcommbee, XBot~dewiki, Chris be, Blaufisch, WarddrBOT, Jperl, BunterElefant, BotMultichill, Der.Traeumer, Christian Stroppel, Ute Erb, VanBot, Nallimbot, Obersachsebot, Xqbot, ArthurBot, Wissens-Dürster, Marzbarz, Zero Thrust, Wondigoma, Ripchip Bot, EmausBot, Chire, Jakob T K, KLBot2, Malvers, Boshomi, Dexbot, TheKatosh, Dgdaniel87 und Anonyme: 53

8.2 Bilder

- **Datei:Commons-logo.svg** *Quelle:* https://upload.wikimedia.org/wikipedia/commons/4/4a/Commons-logo.svg *Lizenz:* Public domain *Autoren:* This version created by Pumbaa, using a proper partial circle and SVG geometry features. (Former versions used to be slightly warped.) *Ursprünglicher Schöpfer:* SVG version was created by User:Grunt and cleaned up by 3247, based on the earlier PNG version, created by Reidab.
- **Datei:Gitter_BW.png** *Quelle:* https://upload.wikimedia.org/wikipedia/commons/7/74/Gitter_BW.png *Lizenz:* CC-BY-SA-3.0 *Autoren:*
- Gitter_BW.jpg *Ursprünglicher Schöpfer:* Gitter_BW.jpg: Maik Außendorf (de:Benutzer:Aussendorf)
- **Datei:Gitter_BW1.png** *Quelle:* https://upload.wikimedia.org/wikipedia/commons/b/b8/Gitter_BW1.png *Lizenz:* CC-BY-SA-3.0 *Autoren:*
- Gitter_BW1.jpg *Ursprünglicher Schöpfer:* Gitter_BW1.jpg: Maik Außendorf (de:Benutzer:Aussendorf)
- **Datei:Gitter_BW2.png** *Quelle:* https://upload.wikimedia.org/wikipedia/commons/6/6d/Gitter_BW2.png *Lizenz:* CC-BY-SA-3.0 *Autoren:*
- Gitter_BW2.jpg *Ursprünglicher Schöpfer:* Gitter_BW2.jpg: Maik Außendorf (de:Benutzer:Aussendorf)
- **Datei:Gitter_BW3.png** *Quelle:* https://upload.wikimedia.org/wikipedia/commons/8/8c/Gitter_BW3.png *Lizenz:* CC-BY-SA-3.0 *Autoren:*
- Gitter_BW3.jpg *Ursprünglicher Schöpfer:* Gitter_BW3.jpg: Original uploader was Aussendorf at de.wikipedia
- **Datei:Gitter_BW4.png** *Quelle:* https://upload.wikimedia.org/wikipedia/commons/9/9b/Gitter_BW4.png *Lizenz:* CC-BY-SA-3.0 *Autoren:*
- Gitter_BW4.jpg *Ursprünglicher Schöpfer:* Gitter_BW4.jpg: Maik Außendorf (de:Benutzer:Aussendorf)
- **Datei:Gitter_BW5.png** *Quelle:* https://upload.wikimedia.org/wikipedia/commons/a/ab/Gitter_BW5.png *Lizenz:* CC-BY-SA-3.0 *Autoren:*
- Gitter_BW5.jpg *Ursprünglicher Schöpfer:* Gitter_BW5.jpg: Maik Außendorf (de:Benutzer:Aussendorf)
- **Datei:SOM_Abb_1.jpg** *Quelle:* https://upload.wikimedia.org/wikipedia/commons/7/76/SOM_Abb_1.jpg *Lizenz:* CC-BY-SA-3.0 *Autoren:* ? *Ursprünglicher Schöpfer:* ?

8.3 Inhaltslizenz

- Creative Commons Attribution-Share Alike 3.0