Vorlesung Computational Intelligence:

Teil 3: Künstliche Neuronale Netze

Radial-Basis-Funktions-Netze (RBF-Netze),
Kohonen-Karten, Kommentare

Ralf Mikut, Wilfried Jakob, Markus Reischl
Karlsruher Institut für Technologie, Institut für Angewandte Informatik
E-Mail: ralf.mikut@kit.edu, wilfried.jakob@kit.edu

jeden Donnerstag 14:00-15:30 Uhr, Nusselt-Hörsaal

Gliederung

Radial-Basis-Funktions-Netze (RBF-Netze)

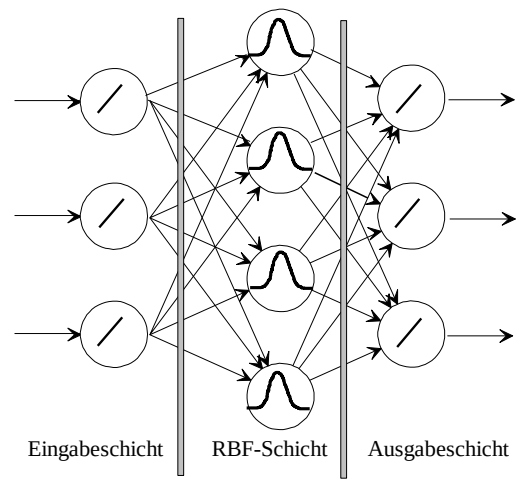
- Idee aus mathematischer Interpolations-/Approximationstheorie
- gegeben:
  - N Datentupel mit
    - bekannten Eingangsgrößen **x**[n], n=1,...,N, **x** ist s-dimensional, und
    - bekannten Ausgangsgrößen y[n], n=1,...,N
  - gesucht: Funktion y = f(**x**), die diese Punkte verbindet
    - Interpolation:
      Funktion läuft GENAU DURCH diese Punkte
    - Approximation/Regression:
      Funktion läuft IN DER NÄHE dieser Punkte weil
      - auf y Störungen liegen (Regression) oder
      - der funktionelle Zusammenhang anders ist (Approximation) oder
      - sowohl Störungen als auch andere funktionelle Zusammenhänge vorliegen (Approgression)
- Eine Lösungstechnik wählt f(.) als Linearkombination von radialsymmetrischen Basisfunktionen

RBF-Netze

Eigenschaften:

- Neuronen in drei Schichten:
  - Eingabeschicht,
  - nur **eine** Schicht mit RBF-Neuronen,
  - Ausgabeschicht
- Verbindungen zwischen den Neuronen vorwärts gerichtet (Feedforward-Netz)
- Funktionen zur Bestimmung des Zustands der RBF-Neuronen sind Gaußfunktionen:

$$z(\mathbf{X},\mathbf{W}) = e^{-\frac{1}{2\sigma^2}\cdot(\mathbf{x}-\mathbf{w})^T(\mathbf{x}-\mathbf{w})}$$
$$= e^{-w_0\cdot(\mathbf{x}-\mathbf{w})^T(\mathbf{x}-\mathbf{w})}$$



Struktur von RBF-Netzen

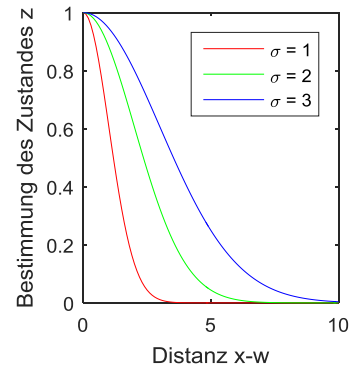
Verdeckte Schicht (RBF-Schicht):

- Parameter w_{ij} kennzeichnen den „Mittelpunkt“ der RB-Funktion (vgl. Mittelwert Normalverteilung)
- Parameter σ (als Teil von w_0) definiert Breite der Funktion (vgl. Streuung der Normalverteilung)
- lineare Aktivierungsfunktion f(z) = z
- für jedes Neuron der RBF-Schicht gilt

$$y_i^{(2)} = e^{-s_i^{(2)}} = \exp\left(- w_{i0}^{(2)} \sum_j (x_j - w_{ij}^{(2)})^2 \right)$$

mit $w_{i0}^{(2)} = \frac{1}{2\sigma^2}$

(σ meist für alle Neuronen gleich, bei Bedarf auch spezifisch für jedes Neuron $\sigma_i^{(2)}$)

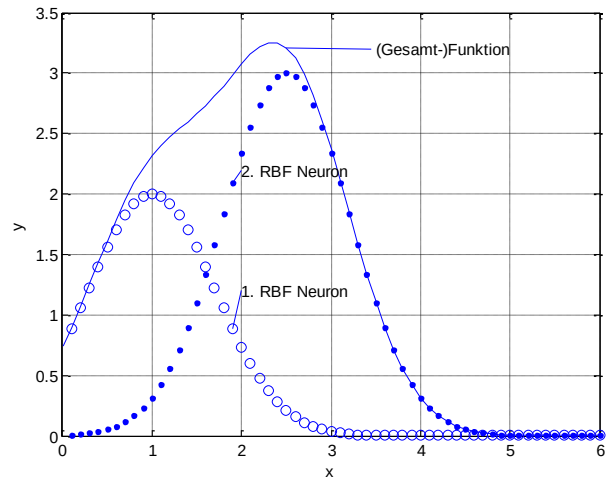


Veranschaulichung Verhalten mit 2 Neuronen

$$y = w_{11}^{(3)} \exp\left(- w_{10}^{(2)} (x - w_{11}^{(2)})^2\right) + w_{12}^{(3)} \exp\left(- w_{20}^{(2)} (x - w_{21}^{(2)})^2\right)$$

Parameter:

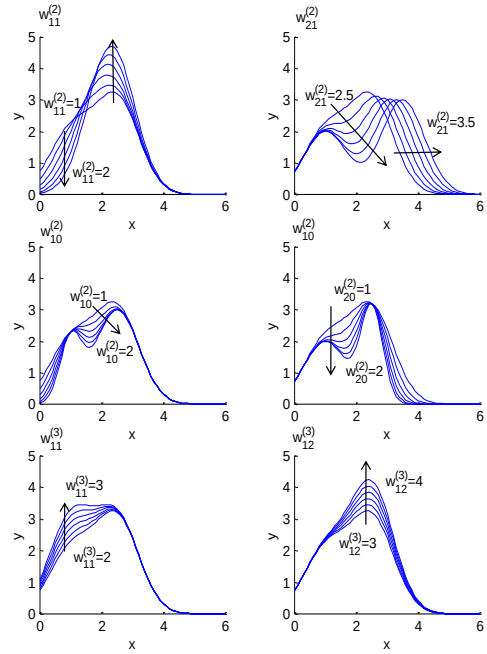
- $w_{11}^{(2)}$ = 1
  Veränderung 1..2
- $w_{21}^{(2)}$ = 2.5
  Veränderung 2.5..3.5
- $w_{11}^{(3)}$ = 2
  Veränderung 2..3
- $w_{12}^{(3)}$ = 3
  Veränderung 3..4
- $w_{10}^{(2)}$ = 1
  Veränderung 1...2
- $w_{20}^{(2)}$ = 1,
  Veränderung 1...2



Wirkung der Parameter

Auswirkung der Veränderungen:

- $w_{11}^{(2)}$ - Gewicht zwischen Neuron Eingabeschicht und 1. RBF-Neuron (oben links)
- $w_{21}^{(2)}$ - Gewicht zwischen Neuron Eingabeschicht und 2. RBF-Neuron (oben rechts)
- $w_{10}^{(2)}$ - Skalierung Einzugsbereich 1.RBF Neuron (mitte links)
- $w_{20}^{(2)}$ - Skalierung Einzugsbereich 2.RBF Neuron (mitte rechts)
- $w_{11}^{(3)}$ - Gewicht zwischen 1. RBF- Neuron und Neuron Ausgabeschicht (unten links)
- $w_{12}^{(3)}$ - Gewicht zwischen 2. RBF-Neuron und Neuron Ausgabeschicht (unten rechts)



Lernen von RBF-Netzen

- Lernen der optimalen Gewichte zum Ausgangsneuron $w_{1j}^{(3)}$
  - mit linearer Aktivierungsfunktion am Ausgangsneuron ergibt sich parameterlineares Schätzproblem
  - kann mit Methode der kleinsten Fehlerquadrate (engl. Least Square) optimal bestimmt werden
- Platzierung der RBF-Neuronen $w_{ij}^{(2)}$
  - nichtlineares Problem
  - mehrere Algorithmen:
    - schrittweise Erhöhung der Anzahl der RBF-Neuronen (z.B. MATLAB-Funktion "newrb.m")
      1. Platzierung eines neuen Neurons am Datentupel mit dem größten Fehler
      2. Lernen der optimalen Gewichte zum Ausgangsneuron für RBF-Netz
      3. Abbruch, wenn Zielgütewerte erreicht, sonst Fortsetzen mit 1.
    - Platzierung auf regelmäßigem Gitter
    - Clusterverfahren (Vorlesung "Datenanalyse für Ingenieure" im Sommersemester)

Kommentare zum Verhalten von RBF-Netzen

Ergebnis:

- Jedes RBF-Neuron der 2. Schicht wird um so stärker angeregt ($f(s) = 0 \ldots 1$), je ähnlicher die Eingangswerte **x** dem jeweiligen Parametervektor **w** sind.
- Verbindungen zwischen RBF-Schicht und Ausgabeschicht wie bei MLP

$\Rightarrow$ Die Verbindung zwischen dem RBF-Neuron, dessen Parametervektor **w** dem Eingangsvektor **x** am nächsten ist, und dem Ausgangsneuron bestimmt maßgeblich den Ausgangswert des Netzes!

$\Rightarrow$ **Jedes Neuron hat nur lokale Wirkung!!!**

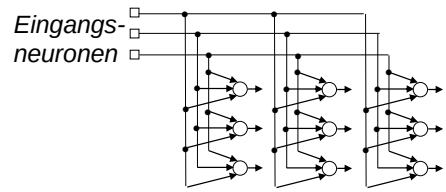
Gliederung

Kohonen-Karten

- Synonyme
  - Self-Organizing Map (SOM)
  - Self-Organizing Feature Map (SOFM)
- Biologische Motivation:
  - Aufbau und Funktion der (menschlichen) Hirnrinde (cerebral cortex)
  - nur ca. 2 mm dick, enthält Milliarden von Neuronen und Hunderte von Milliarden von Synapsen
  - Bereiche können bestimmten senso-motorischen Funktionen zugeordnet werden, z. B. dem Hören, dem Sehen, der Motorik.
  - Ähnliche Muster werden auf räumlich benachbarte Neuronen abgebildet
- Hieraus wurde durch Kohonen das Prinzip der Bildung topografischer Karten abgeleitet:
  Die räumliche Lage eines Ausgangsneurons in der topografischen Karte entspricht einem Bereich oder einem Merkmal in den Eingangsdaten.
- Durch SOM werden Eingangsdaten mit hoher Dimension (viele Einzelmerkmale) häufig auf Karten niederer Dimension (meist ein- oder zweidimensional) abgebildet.

Aufbau und Funktionsweise von SOMs

- Entsprechend dem biologischen Vorbild werden in einem SOM die Ausgangsneuronen in einem ein- oder zweidimensionalen Gitter angeordnet.
- Feedforward-Netz
- Der Eingangsvektor **x** hat s Komponenten. Dementsprechend hat der Parametervektor des *j*-ten Ausgangsneurons $\mathbf{w}_j$ die Dimension s.
- Wird dem SOM ein Eingangsvektor vorgelegt, wird bei einem angelernten Netz lediglich eine Gruppe benachbarter Ausgangsneuronen aktiviert.

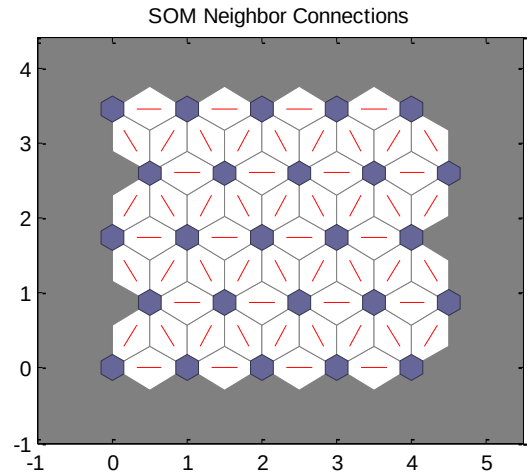


Zweidimensionales Gitter der Ausgangsneuronen

- Berechnung des Zustands:
  - über Distanz, meist Verwendung des Euklidischen Abstands: $z_j=||\mathbf{x}-\mathbf{w}_j||$
  - Alternative: Verwendung des inneren Produkts bei normalisierten Gewichtsvektoren **w** und Eingangsvektoren **x**: $z_j=\mathbf{x}^T\mathbf{w}_j$
- Ausgang des Netzes: Gewinnerneuron $j(\mathbf{x})$ (*winner-takes-all*-Prinzip) ergibt sich mit $j(\mathbf{x})=\text{argmin } z_j$.

Nachbarschaften

- Nachbarschaftsstruktur ist fest vorgegeben
- Beispiel:
  - zweidimensional
  - je 5 Neuronen pro Dimension
  - Neuronen in der Mitte haben 6 Nachbarn
  - Neuronen am Rand haben 2-4 Nachbarn



Lernverfahren für SOMs

1. Festlegen der Dimension des Gitters und der Anzahl der Ausgangsneuronen
2. zufällige Initialisierung der Gewichte $\mathbf{w}_{SOM,i}[0]$
3. zufällige Auswahl eines Datentupels $\mathbf{x}[n]$
4. Bestimmung des Gewinnerneuron wird für dieses Datentupel bestimmt
5. Bestimmung der Nachbarn für das Gewinnerneuron
6. Gewinnerneuron (besonders stark) und dessen Nachbarn (etwas weniger) werden in Richtung von x[n] verschoben, k: Iterationsschritt:

$$\mathbf{w}_{SOM,i}[k+1] = \mathbf{w}_{SOM,i}[k] + \rho_{i,i_G[k]}[k](\mathbf{x}[k] - \mathbf{w}_{SOM,i}[k])$$

$$i_G[k] = \mathrm{argmin}_i\, d(\mathbf{w}_{SOM,i}[k], \mathbf{x}[k])$$

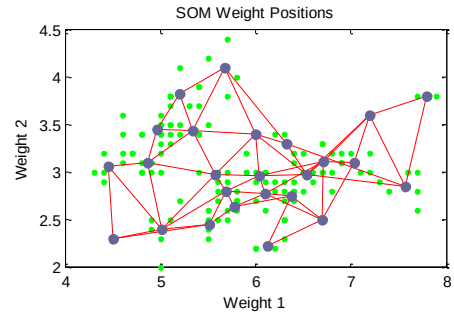
$$\rho_{i,j}[k] = \rho_0[k] \cdot \exp(-d(\mathbf{p}_i, \mathbf{p}_j)) \text{ mit } \rho_{i_G[k],i_G[k]}[k] = \rho_0[k] \geq \rho_{i,i_G}[k]$$

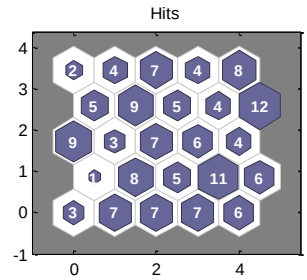
7. Berechnung eines Gütekriteriums (basierend auf dem durchschnittlichen Abstand der letzten Datentupel zum Gewinnerneuron)
8. Abbruch, wenn Güteanforderung erfüllt, sonst k=k+1 und Fortsetzen mit 3.

Erweiterungen möglich, z.B. Änderung Nachbarschaft über k usw.

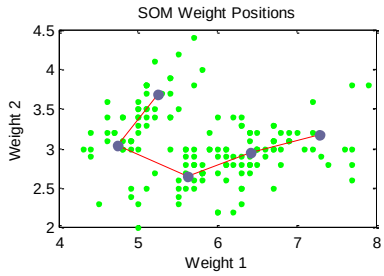
Ergebnisse

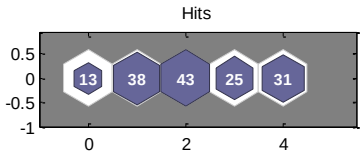
- Neuronen werden in die Nähe von Datentupeln gezogen
- Gewichte $\mathbf{w}_{SOM,i}$ werden so bestimmt, dass sie wichtige Bereiche des Eingangsraum abdecken
- 1D- bzw. 2D-Verbindungsstruktur bleibt erhalten: "Topologieerhaltende Abbildung"
- projiziert bei Bedarf höherdimensionale auf niedrigdimensionale Eingangsräume
- Ergebnis kann im Bereich der Verbindungsstruktur analysiert werden, z.B. wieviele Datentupel pro Neuron (siehe Beispiel)

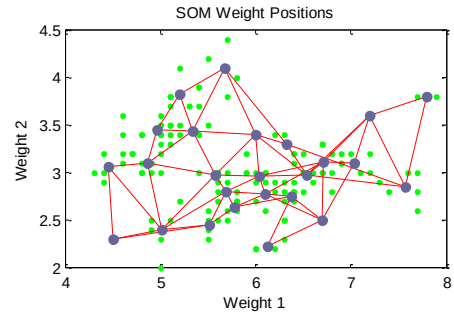


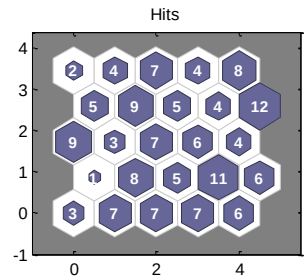


1D- und 2D-SOMs







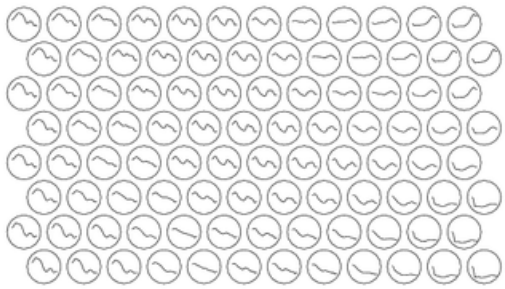


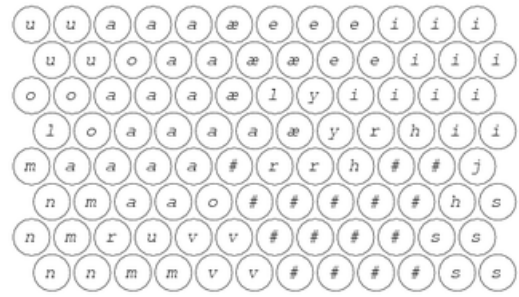
Anwendungen für SOMs

- *Problemtypen*
  - Dimensionsreduktion bei höherdimensionalen Merkmalsräumen: z. B. in der Spracherkennung, Klassifizierung von Phonemen (Transformation der durch Fourier-Transformation gewonnen Merkmale auf zweidimensionale Phonemkarte)
  - Approximation und Visualisierung von höherdimensionalen nichtlinearen Zusammenhängen: z. B. Robotersteuerungen
- *Anwendungsfelder*
  - Sprachverarbeitung
  - Bildverarbeitung
  - Robotik/autonome Systeme
  - Telekommunikation

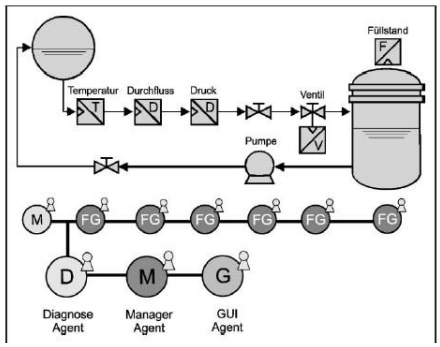
Beispiel: Spracherkennung

- Finnische Phonemkarte
  Bildquelle: http://www.scholarpedia.org/article/Kohonen_network
  (Teuvo Kohonen)
  Links: Spektrum
  Rechts: Phoneme (# Stoppkonsonanten k,p,t)





Beispiel: Visualisierung von Prozessphasen (1)



Quelle:

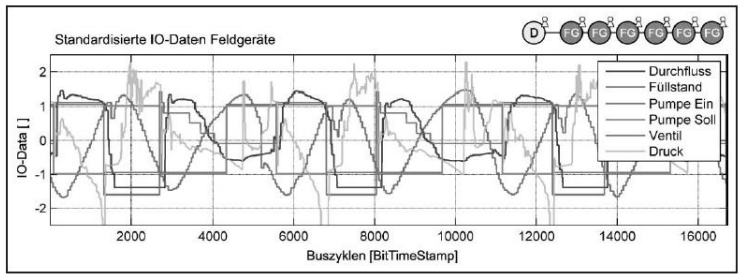
Frey, C.: Prozessdiagnose und Monitoring feldbusbasierter Automatisierungsanlagen mittels selbstorganisierender Karten und Watershed-Transformation. *at-Automatisierungstechnik, Oldenbourg,* **2008***, 56*, 374-380

Eingang:

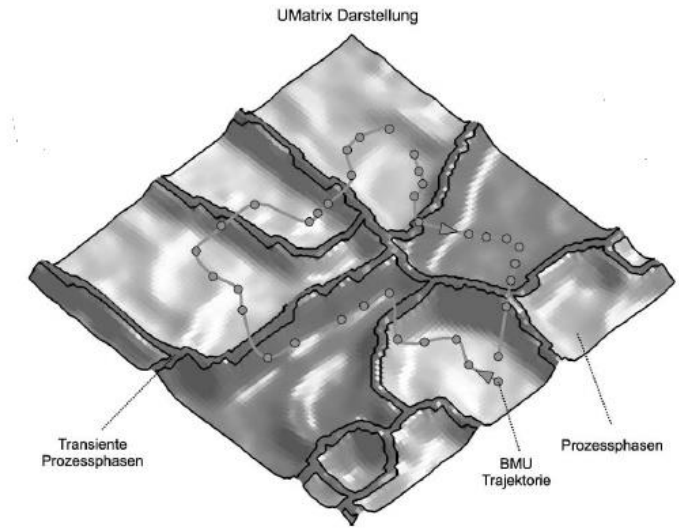
Rohdaten aus einem Prozess

Ausgang:

Visualisierung via SOM

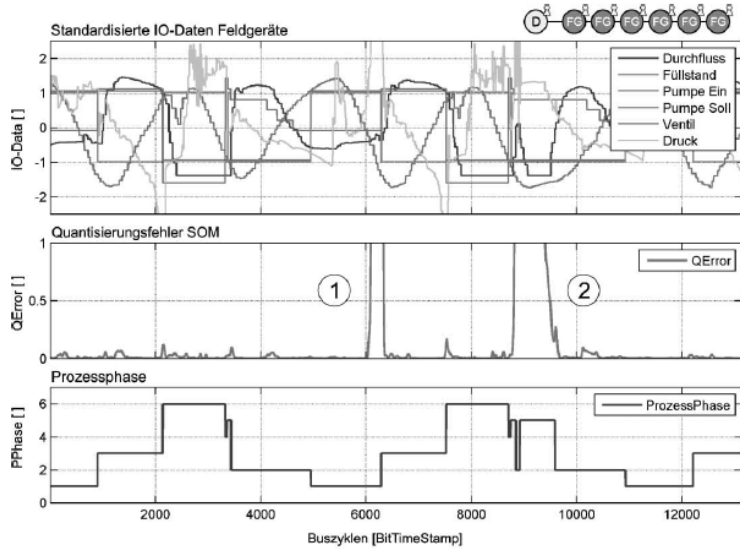


Beispiel: Visualisierung von Prozessphasen (2)



Frey, C.: Prozessdiagnose und Monitoring feldbusbasierter Automatisierungsanlagen mittels selbstorganisierender Karten und Watershed-Transformation. *at-Automatisierungstechnik, Oldenbourg,* **2008***, 56*, 374-380

Beispiel: Visualisierung von Prozessphasen (3)



Frey, C.: Prozessdiagnose und Monitoring feldbusbasierter Automatisierungsanlagen mittels selbstorganisierender Karten und Watershed-Transformation. *at-Automatisierungstechnik, Oldenbourg,* **2008**, *56*, 374-380

Gliederung

Kommentare

- Künstliche Neuronale Netze sind wegen ihrer Eigenschaft als "universelle Approximatoren" populär
- dann besonders sinnvoll, wenn strukturelle Zusammenhänge unbekannt sind
- Künstliche Neuronale Netze sind trotzdem eher kompliziert, deshalb unbedingt mit einfacheren Methoden (Polynomregression usw.) vergleichen
- MLP, RBF und SOM sind die Standardtypen für Künstliche Neuronale Netze
- Neuere Trends existieren, z.B. Deep Learning
  - Netze mit vielen verdeckten Schichten
  - z.T. speziell designte Schichten, z.B. bei Bildverarbeitung
    - Downsampling (Verringerung der Auflösung)
    - fest platzierte lokale Elemente
      (werden auch als Convolutional Neural Networks bezeichnet)
- Rekurrente Netze stark umstritten
  - Pro: Abbildung dynamischer Zusammenhänge
  - Kontra: Konvergenzprobleme

Software

- Software
  "Comparison of Neural Network Simulators" (U Colorado)
  https://grey.colorado.edu/emergent/index.php/Comparison_of_Neural_Network_Simulators

- MATLAB:
  - Neuronale Netze Toolbox inkl. MLP, RBF, SOM usw., Hilfe mit
    >> help nnet
  - Gait-CAD Implementierungen
    - MLP, RBF, SOM:
      (frei verfügbar unter http://sourceforge.net/projects/gait-cad/)
    - Projekte im ILIAS:
      1D, 2D Regression (künstliche Testdatensätze)
    - Projekte in Gait-CAD:
      Building-Datensatz für Energieverbrauch in einem Gebäude

Ankündigung Übung Computational Intelligence

- Zeit und Ort:
  - Dienstag, 15.12.2015 14.00 - 15.30 Uhr ODER 15:45-17:15 im SCC Geb. 20.20 H-Pool (je 29 PCs)
  - Einschreibelisten im ILIAS
- Ziele:
  - Kennenlernen MATLAB-Toolbox Gait-CAD
  - Anlernen von MLPs im ein- und zweidimensionalen Fall
  - Zusatzaufgabe: Regression Energieverbrauch Gebäude

ACHTUNG:

- Für die Übung wird der Rechenzentrumslogin für jeden Studenten benötigt
- Material
  - Aufgabenstellung inkl. Installationshinweise Gait-CAD
  - Beispieldatensätze und Makros
- Download unter ILIAS

Bedienung Gait-CAD

- Funktionen über Menüs zugänglich (Callbacks...)
- Parameter über Oberfläche einstellbar (Callbacks...)
- Ergebnisse in Bilder und Dateien
- Automatisierung von Abläufen durch Makros
- Einbindung neuer Funktionen:
  - Makro
  - Plugin Merkmalsextraktion
  - Menüpunkt usw.
- Installationsdatei
- Handbuch (>170 Seiten)