Was bisher geschah

Künstliche Neuronen:

- Mathematisches Modell und Funktionen:
  Eingabe-, Aktivierungs- Ausgabefunktion
- Boolesche oder reelle Ein-und Ausgaben
- Aktivierungsfunktionen:
  - Schwellwertfunktion
  - lineare Funktion
  - sigmoide Funktion

Künstliche Neuronale Netze:

- Aufbau: gerichteter Graph mit Kantengewichten (Gewichtsmatrix)
- Feed-Forward-Netze
- Training (schrittweise Minimierung der quadratischen Abweichung auf der Trainingsmenge):
  - Δ-Regel für Ein-Schicht-Feed-Forward-Netze mit linearer oder Schwellwert-Aktivierung
  - Backpropagation für Mehr-Schicht-Feed-Forward-Netze mit sigmoider Aktivierung

Radiale-Basisfunktions-Netze

Anwendung zur Klassifizierung von Mustern (Merkmalsvektoren)

Annahmen:

- Klassen haben Zentren (Schwerpunkte),
- alle Eingabevektoren nahe dazu gehören zur selben Klasse

2-Schicht-FFN mit vollständig verbundenen Schichten

- Eingaben $x \in \mathbb{R}^m$
- Ausgaben $y \in \mathbb{R}^n$
- eine versteckte Schicht h (mit l Neuronen)
  enthält oft mehr Neuronen als die Eingabeschicht

Neuronen der verschiedenen Schichten haben verschiedene Aktivierungsfunktionen:

- versteckte Schicht: nichtlinear
- Ausgabeschicht: linear

Netz berechnet eine Funktion $f : \mathbb{R}^m \to \mathbb{R}^n$

Versteckte Neuronen im RBF-Netz

Idee:

- Eingangsgewichte eines Neurons j der versteckten Schicht interpretiert als Koordinaten eines Punktes $(w_{1j}, \ldots, w_{mj}) \in \mathbb{R}^m$ (Zentrum einer Klasse)
- Eingangsfunktion $I_j : R^m \to \mathbb{R}$ des Neurons j berechnet Abstand des Eingabevektors $(x_1, \ldots, x_m)$ vom Zentrum $(w_{1j}, \ldots, w_{mj}) \in \mathbb{R}^m$
- Aktivierungfunktion: radiale Basisfunktion $A_j : \mathbb{R} \to \mathbb{R}$ nimmt größten Wert im Zentrum an fällt mit wachsendem Abstand vom Zentrum
- das Neuron der versteckten Schicht am aktivsten, welches das zum Eingabevektor nächste Zentrum repräsentiert

Abstandsfunktionen

(Eingabefunktionen der versteckten Neuronen im RBF-Netz)

Abstandsfunktion $d : \mathbb{R}^{2m} \to \mathbb{R}$ mit den Eigenschaften:

- $\forall x, y \in \mathbb{R}^m : d(x, y) = 0$ gdw. $x = y$
- $\forall x, y \in \mathbb{R}^m : d(x, y) = d(y, x)$ (kommutativ)
- $\forall x, y, z \in \mathbb{R}^m : d(x, y) + d(y, z) \geq d(x, z)$ (Dreiecksungleichung)

Beispiele: $I(x_1, \ldots, x_m) = d_k(x, w_j) = \sqrt[k]{\sum_{k=1}^{m} (w_{kj} - x_k)^k}$

- für $k = 2$: $I(x_1, \ldots, x_m) = d_2(x, w_j) = \sqrt{\sum_{k=1}^{m} (w_{kj} - x_k)^2}$ Euklidischer Abstand zwischen Eingangs- und Gewichtsvektor
- für $k = 1$: $I(x_1, \ldots, x_m) = d_1(x, w_j) = \sum_{k=1}^{m} |w_{kj} - x_k|$ Manhattan-Metrik
- für $k \to \infty$: $I(x_1, \ldots, x_m) = \max\{|w_{kj} - x_k| \mid i \in \{1, \ldots, m\}\}$ Maximum-Metrik

Radiale Funktionen

Radiale Funktion $f : \mathbb{R}_{\geq 0} \to [0, 1]$ mit den folgenden Eigenschaften:

- aus $x < y$ folgt $f(x) \geq f(y)$ (monoton fallend)
- $f(0) = 1$
- $\lim_{x \to \infty} f(x) = 0$ (verschwindet im Grenzwert)

(fällt ausgehend vom Zentrum 0 in alle Richtungen)

Beispiele:

- Schwellwertfunktion (fallend)

$$f_\theta(x) = \begin{cases} 0 & \text{falls } x > \theta \\ 1 & \text{sonst} \end{cases}$$

- linear $f_m(x) = \max(0, 1 - mx)$
- Gauß-Funktion $f_c(x) = e^{-cx^2}$

Ausgabeneuronen im RBF-Netz

- Eingaben (von der versteckten Schicht): $h \in \mathbb{R}^l$
- Gewichte: $W' \in \mathbb{R}^{l \times n}$
- Ausgaben: $y \in \mathbb{R}^n$

- Eingabefunktion: gewichtete Summe
- Aktivierungsfunktion: Identität (linear)
- Ausgabefunktion: Identität

(Schwellwertneuronen mit linearer Aktivierung)

RBF-Netze: Beispiele

- 2-1-1 -Netz für $\wedge$
  - erste Schicht (RBF): Zentrum $w_{1,h} = w_{2,h} = 1$,
    Eingabefunktion: Euklidische Metrik
    Aktivierung: Stufenfunktion
    Radius $\theta_h = 1/2$
  - zweite Schicht: Gewicht $w_{h,y} = 1$,
    Eingabefunktion: gewichtete Summe
    Aktivierung: linear
    Schwellwert $\theta_y = 0$
- 2-2-1-Netz für $\leftrightarrow$:
  Idee: $x_1 \leftrightarrow x_2 \equiv (x_1 \wedge x_2) \vee \neg(x_1 \vee x_2)$
  - erste Schicht (RBF): Zentren $w_{1,h1} = w_{2,h1} = 1$,
    $w_{1,h2} = w_{2,h2} = 0$,
    Eingabefunktion: Euklidische Metrik
    Aktivierung: Stufenfunktion
    Radien $\theta_{h1} = \theta_{h2} = 1/2$
  - zweite Schicht: Gewichte $w_{h1,y} = w_{h2,y} = 1$,
    Eingabefunktion: gewichtete Summe
    Aktivierung: linear
    Schwellwert $\theta_y = 0$

RBF-Netze zur Approximation von Funktionen

Approximation einer Funktion $f : \mathbb{R} \to \mathbb{R}$ durch Linearkombination (gewichtete Summe) von radialen Funktionen, z.B.

- stückweise konstante Funktionen (Stufen)
- stückweise lineare Funktionen
- Gauß-Funktionen

Zwei-Schicht-FF-Netz:

- ein Eingabeneuron x
- k versteckte Neuronen $h_1, \ldots, h_k$
  jedes für eine Basisfunktion
- ein Ausgabeneuron y

Beispiel

Approximation n-stelliger Boolescher Funktionen:

- n Eingabeneuronen x_i
- 2^n versteckte Neuronen h_i
  Eingangsgewichte (jede mögliche Eingabe als Zentrum)
  Eingangsfunktion: Euklidische oder Manhattan-Metrik
  Aktivierung: Stufenfunktion
  alle Radien $1/2$
- ein Ausgabeneuron y
  zu bestimmende Gewichte w_i, Schwellwert 0

RBF-Netze – Lernen

übliches Vorgehen: nacheinander

1. Gewichte der ersten Schicht
   (Eingabe zu versteckten Neuronen):
   Bestimmung der Anfangspunkte der Zentren, z.B.
   - gleichmäßig überdeckend
   - alle Trainingsmuster
   - durch zufällige Auswahl von Trainingsmustern
   - durch Clustering-Techniken,
     z.B. unüberwachtes Training (später)
2. Gewichte der zweiten Schicht (zu Ausgabeneuronen):
   direkte Berechnung oder überwachtes Training
   (z.B. Delta-Regel)
   Bestimmung der Faktoren vor den Basisfunktionen

Eigenschaften von RBF-Netzen

Vorteile:

- einfache Topologie
- schnelle Berechnung
- Netzausgabe außerhalb der Trainingsmenge gering
- Gewichte können direkt bestimmt werden (ohne Training)

Nachteile:

- Qualität der Approximation durch Lage der Zentren bestimmt
- Lernerfolg hängt stark von der Start-Instanziierung der Gewichte der ersten Schicht (Zentren) ab
- Auswendiglernen der Trainingsdaten

Cognitron

Beobachtungen im visuellen System:

- sendet vorverarbeitete Signale an Gehirn
- Verbindung benachbarter Neuronen
  horizontale Zellen berechnen Mittelwert (der Helligkeit)
  wirken hemmend auf Signale nahe beim Mittelwert
- ähnlich Faltung in DBV

Bild-Pyramiden

Features:

- Flächen gleicher Farbe
- Kanten
- Formen
- Texturen, ...

Idee aus DBV:
Bilder enthalten Informationen auf verschiedenen Ebenen,
kleinteilige Beobachtung lenkt evtl. von wesentlichen Merkmalen ab
Umsetzung durch Multiskalen-Bilder (Pyramiden)
entstehen durch mehrfache Wiederholung von

- Glättung (durch geeignete Filter)
- Komprimierung durch geringere Abtastrate,
  z.B. Gauß-Pyramide: Löschen jeder zweiten Zeile und Spalte

Umsetzung als KNN (feed-forward)

Neocognitron

Fukushima, 1975: Cognitron: A Self-Organizing Multilayered Neural Network Model

1983: Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition

Motivation: Erkennung handschriftlicher Ziffern

Aufbau Neocognitron:

- Eingabe-Schicht
- vier (oder mehr) versteckte Stufen aus je zwei Schichten:
  1. Transformation in 12 Bilder (Ebenen)
     Feature-Extraktion (Faltungen mit je einem 3×3-Kern)
     Filterkerne durch Eingangsgewichte definiert (weight sharing)
     Gewichte durch Trainingsmuster gelernt
  2. Kombination mehrerer transformierter Bilder
     z.B. punktweise gewichtete Summe, Max
     Gewichte nicht trainiert
- Ausgabe nach letzter Kombinations-Schicht (Klassifikation)
- inkrementelles Lernen stufenweise von Ein- zu Ausgabeschicht

mehrere Varianten mit überwachtem und unüberwachtem Lernen

Convolutional Neural Networks

z.B. Alex Krizhevsky, . . . , 2012:
ImageNet Classification with Deep Convolutional Neural Networks
prinzipieller Aufbau:

- Eingabe-Schicht
- Versteckte Stufen aus je mehreren Schichten
  - Faltungs-Schicht (Feature-Maps)
  - evtl. ReLU-Schicht (nichtlinear)
  - gelegentlich Subsampling-Schicht (Pooling)

  mehrfache Wiederholung (deep), evtl. in verschiedenen Reihenfolgen
- evtl. klassische Schichten mit vollständigen Verbindungen zwischen benachbarten Schichten
- Ausgabe-Schicht

inzwischen auch komplexere Konstruktionen, z.B.

- AlexNet (Dropout-Schichten)
- GoogLeNet (Inception)
- ResNet (skip connections)

CNN-Lernen

Überwachtes Lernen durch Backpropagation:

- Faltungsschichten:
  Backpropagation durch Faltung mit gespiegelten Kernen
- Pooling-Schichten (z.B. bei Max-Pooling):
  auf Hinweg Position (Koordinaten) des maximalen Elementes speichern
  Backpropagation: Abstieg in Richtung dieser Position
- klassische Schichten: Gradientenabstieg wie bisher