Thema 3: Radiale Basisfunktionen und RBF-Netze

Barbara Rakitsch

Zusammenfassung:
Aufgabe dieses Vortrags war es, die Grundlagen der RBF-Netze darzustellen.

1 Einführung

RBF-Netze sind eine spezielle Form von künstlichen neuronalen Netzen. Im Gegensatz zu den bisher betrachteten Netzen werden nur Punkte klassifiziert, die sich in der Nähe der Zentren der radialsymmetrischen Aktivierungsfunktionen befinden.

1.1 Idee der künstlichen neuronalen Netze

Künstliche neuronale Netze simulieren stark vereinfacht biologische neuronale Netze.

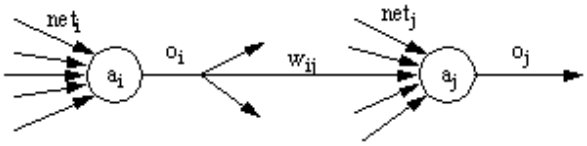


Abbildung 1: Zellen eines neuronalen Netzes

Sie bestehen aus

- Zellen:
  Sie besitzen einen Aktivierungszustand $a_i(t)$. Dieser berechnet sich aus der Aktivierungsfunktion $a_j(t+1) = f_{act}(a_j(t), net_j(t), \theta_j)$, wobei $net_j(t)$ die Propagierungsfunktion und θ_j den Schwellenwert des Neurons j bezeichnet. Zusätzlich besitzen sie eine Ausgabefunktion $o_j = f_{out}(a_j)$. Dadurch wird festgelegt, welche Information ein Neuron an seine Nachfolgeneuronen weitergibt.

- Verbindungsnetzwerk:
  Die Neuronen sind über gerichtete und gewichtete Kanten, w_{ij}, miteinander verbunden. Damit wird festgelegt, in welche Richtung Informationen weitergeleitet werden können, und wie wertvoll diese Informationen sind.

- Propagierungsfunktion:
  Sie berechnet die Netzeingabe eines Neurons j anhand folgender Formel:
  $net_j(t) = \sum_i o_i(t) w_{ij}$

2 Definition der RBF-Netze

RBF-Netze sind vorwärts gerichtete neuronale Netze. Die Verbindungen verlaufen von den Eingabe- in Richtung der Ausgabeneuronen.
Sie besitzen nur eine Schicht von verdeckten Neuronen. In dieser Schicht findet die Informationsverarbeitung statt.

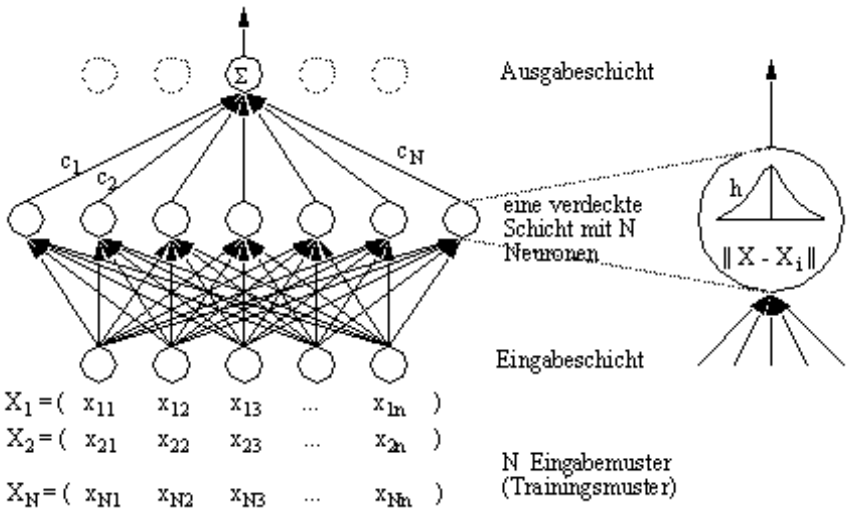


Abbildung 2: Struktur von RBF-Netze

Die Anzahl der Neuronen in der Eingabeschicht entspricht der Dimension der Eingabevektoren. Über jedes Eingabeneuron wird also eine Komponente des Eingabevektors in das Netz geladen.

Die Aktivierungsfunktionen der Neuronen in der verdeckten Schicht sind aus radialsymmetrischen Basisfunktionen aufgebaut: $h_i(\|X - X_i\|)$, wobei X der Eingabevektor und X_i der Zentrumsvektor ist. Damit ist $\|X - X_i\|$ ein skalarer, positiver Wert.

h_i liefert große Werte, wenn die zwei Vektoren nahe zusammen liegen und dementsprechend kleine Werte, wenn sie weit voneinander entfernt sind.

Die Aktivierungsfunktionen der Neuronen in der verdeckten Schicht sind gleichzeitig die Ausgabefunktionen dieser Neuronen. Über die Gewichtskoeffizienten c_i wird der Einfluss des Neurons i auf das Endergebnis festgelegt.

Damit berechnet ein RBF-Netz folgende Funktion:
$f : \mathbb{R}^n \rightarrow \mathbb{R} : f(X) = \sum_{i=1}^{N} c_i h_i(\|X - X_i\|)$

3 Interpolation mit Zentrumsfunktionen

Das Netz erhält ein Trainingsmuster mit N Eingabevektoren $X_1, \cdots, X_N$ und den dazugehörigen Ausgabewerten $y_1, \cdots, y_N$. Die Ausgabe des Netzes ist eine Funktion, für die gilt:
$f(X_i) = y_i, \forall i = 1, \cdots, N$

In der verdeckten Schicht befinden sich genau N Neuronen. Die Zentrumsvektoren dieser Neuronen entsprechen den Eingabevektoren der Trainingsmenge.

Sind die Funktionen h_i bekannt, können die Koeffizienten c_i über ein lineares Gleichungssystem bestimmt werden:

$$
\begin{aligned}
&c_1 h_1(\|X_1 - X_1\|) + c_2 h_2(\|X_1 - X_2\|) + \cdots + c_N h_N(\|X_1 - X_N\|) = y_1 \\
&c_1 h_1(\|X_2 - X_1\|) + c_2 h_2(\|X_2 - X_2\|) + \cdots + c_N h_N(\|X_2 - X_N\|) = y_2 \\
&\vdots \\
&c_1 h_1(\|X_N - X_1\|) + c_2 h_2(\|X_N - X_2\|) + \cdots + c_N h_N(\|X_N - X_N\|) = y_N
\end{aligned}
$$

In Vektorschreibweise:

$$
C = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix}, Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}, H = \begin{pmatrix} h_1(\|X_1 - X_1\|) & h_2(\|X_1 - X_2\|) & \cdots & h_N(\|X_1 - X_N\|) \\ h_1(\|X_2 - X_1\|) & h_2(\|X_2 - X_2\|) & \cdots & h_N(\|X_2 - X_N\|) \\ \vdots & \vdots & \vdots & \vdots \\ h_1(\|X_N - X_1\|) & h_2(\|X_N - X_2\|) & \cdots & h_N(\|X_N - X_N\|) \end{pmatrix}
$$

Damit lässt sich obige Formel umschreiben in:
$HC = Y$

Die gesuchten Koeffizienten sind durch Umformen der Gleichung gegeben:
$C = H^{-1}Y$ $_{(N \times 1) = (N \times N)(N \times 1))}$

3.1 Verrauschte Eingabedaten

Sind die Eingabedaten verrauscht, muss das Verfahren modifiziert werden:
$C = (H + \lambda \mathbb{I})^{-1} Y$,
der Faktor λ ist proportional zu dem Rauschen, und $\mathbb{I}$ steht für die Einheitsmatrix.

Das Interpolationsproblem wird somit approximierend gelöst. Geht λ gegen Null, wird das Interpolationsproblem wieder genau gelöst.

3.2 Gebräuchliche Zentrumsfunktionen

- $h(z) = e^{-(\frac{z}{\alpha})^2}$,

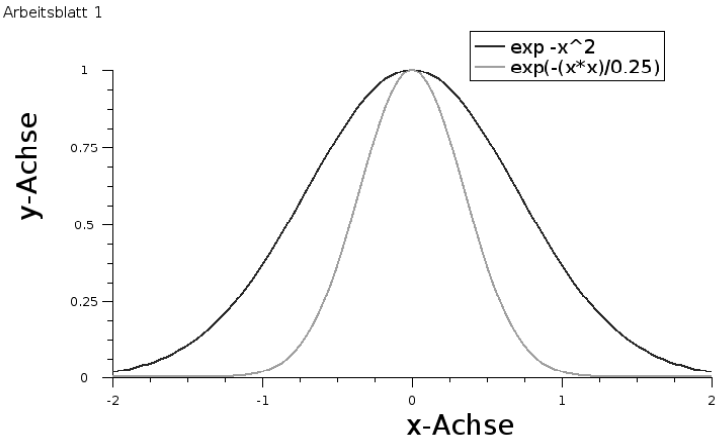


Abbildung 3: modifizierte Gauß-Funktionen

α definiert die Breite der Funktion. Je kleiner α ist, um so schmaler ist die Funktion.

- $h(z) = \frac{1}{(c^2+z^2)^\alpha}$,
- $h(z) = \sqrt{z^2 + c^2}$,

3.3 Beispiel einer Interpolation

Es ist eine Funktion gesucht, die durch die Punkte

- $f(1) = 1$
- $f(2) = e^{-3}$

verläuft.

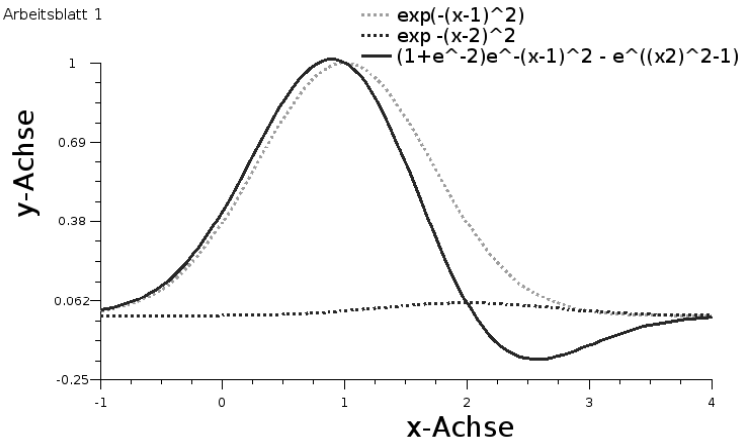
Verwendete Zentrumsfunktion: $h(z) = e^{-z^2}$

Rechnung:

$$H = \begin{pmatrix} h_1(\|X_1 - X_1\|) & h_2(\|X_1 - X_2\|) \\ h_1(\|X_2 - X_1\|) & h_2(\|X_2 - X_2\|) \end{pmatrix} = \begin{pmatrix} h_1(0) & h_2(1) \\ h_1(1) & h_2(0) \end{pmatrix}$$

$$C = H^{-1}Y = \begin{pmatrix} 1 & e^{-1} \\ e^{-1} & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ e^{-3} \end{pmatrix} \Leftrightarrow$$
$$C = \frac{1}{1-e^{-2}} \begin{pmatrix} 1 & -e^{-1} \\ -e^{-1} & 1 \end{pmatrix} \begin{pmatrix} 1 \\ e^{-3} \end{pmatrix} \Leftrightarrow$$
$$C = \frac{1}{1-e^{-2}} \begin{pmatrix} 1 - e^{-4} \\ -e^{-1} + e^{-3} \end{pmatrix}$$
$$C = \begin{pmatrix} 1 + e^{-2} \\ -e^{-1} \end{pmatrix}$$

Lösung:
$f(x) = (1 + e^{-2})e^{-(x-1)^2} - e^{(x-2)-1}$



3.4 Nachteile einer Interpolation

1. Die Funktion ist nur durch die bekannten Trainingsmuster festgelegt. Dazwischen kann die Funktion beliebig stark schwanken. Oft ist ein glatter Kurvenverlauf jedoch von Vorteil.

2. Beruhen die Eingabedaten auf Messergebnissen, ist eine exakte Interpolation nicht sinnvoll, da die Daten von vornherein mit Fehlern behaftet sind.

3. Je mehr Trainingsmuster verwendet werden, um so größer ist das RBF-Netz, da die Neuronenzahl in der verdeckten Schicht genau der Anzahl der Trainingsmuster entspricht. Dies hat einen höheren Rechenaufwand für die Bestimmung der Gewichtskoeffizienten zur Folge. Bei großangelegten Versuchsreihen wird dieses Verfahren damit zunehmend ineffizient.

4 Approximation mit Zentrumsfunktionen

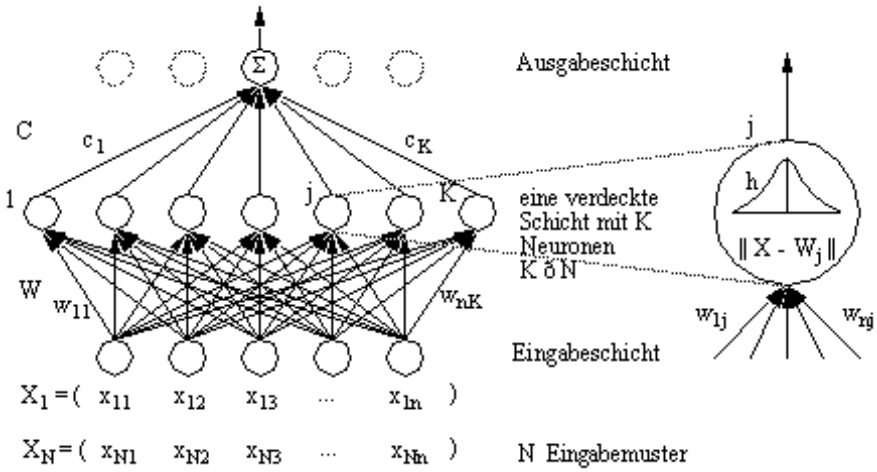


Abbildung 4: Struktur von RBF-Netzen

Die Anzahl K der Neuronen in der verdeckten Schicht ist konstant, mit $K \leq N$. Als Zentren werden oft Untermengen der Trainingsmuster verwendetet (siehe Kapitel 6).

Das Netz berechnet folgende Funktion:
$f : \mathbb{R}^n \to \mathbb{R} : f(X) = \sum_{i=1}^{K} c_i h_i(\|X - X_i\|)$

Will man die Gewichtskoeffizienten bestimmen, muss man ein lineraes Gleichungssystem mit K Unbekannten und N Gleichungen lösen:

$$C = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_K \end{pmatrix}, Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}, H = \begin{pmatrix} h_1(\|X_1 - X_1\|) & h_2(\|X_1 - X_2\|) & \cdots & h_K(\|X_1 - X_K\|) \\ h_1(\|X_2 - X_1\|) & h_2(\|X_2 - X_2\|) & \cdots & h_K(\|X_2 - X_K\|) \\ \vdots & \vdots & \vdots & \vdots \\ h_1(\|X_N - X_1\|) & h_2(\|X_N - X_2\|) & \cdots & h_K(\|X_N - X_K\|) \end{pmatrix}$$

Es handelt sich also um ein Gleichungssystem mit mehr Gleichungen als Unbekannten. Dies ist nur lösbar, wenn $N - K$ Gleichungen voneinander linear abhängig sind.

Sonst kann keine Funktion gefunden werden, die die Eingabedaten interpoliert. Es wird nun eine Funktion gesucht, die den Gesamtfehler $E = \sum_{i=1} N(y_i - o_i(X))^2$ minimiert, wobei y_i für den gewünschten und $o(X_i)$ für den tatsächlichen Ausgabewert des Netzes steht.

Dazu wird die Moore-Penrose Pseudoinverse benötigt:
$H^+ = (H^T H)^{-1} H^T$
$H^+ = ((N \times K)^T (N \times K))^{-1} (N \times K)^T = ((K \times N)(N \times K))^{-1} (K \times N) = (K \times K)(K \times N) = (K \times N)$

Lösung des Approximationsproblems:
$C = H^+ Y$
$(C = (K \times N)(N \times 1)) = (K \times 1)$

4.1 Beispiel einer Approximation

Es ist eine Funktion gesucht, die durch die Punkte

- $f(1) = 1$, und $f(2) = e^{-3}$

verlaufen soll.

Es steht jedoch nur ein Neuron in der verdeckten Schicht zur Verfügung.
Als Zentrum des Neurons wird $x_1 = 1$ gewählt.

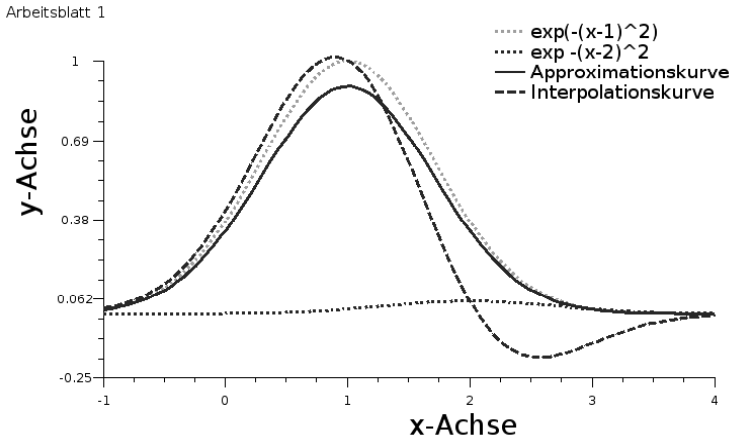
Verwendete Zentrumsfunktion: $h(z) = e^{-z^2}$

Rechnung:

$$H^+ = (H^T H)^{-1} H^T = \left(\begin{pmatrix} 1 & e^{-1} \end{pmatrix} \begin{pmatrix} 1 \\ e^{-1} \end{pmatrix} \right)^{-1} \begin{pmatrix} 1 & e^{-1} \end{pmatrix} = \frac{1}{1+e^{-2}} \begin{pmatrix} 1 & e^{-1} \end{pmatrix}$$

$$C = H^+ Y = \frac{1}{1+e^{-2}} \begin{pmatrix} 1 & e^{-1} \end{pmatrix} \begin{pmatrix} 1 \\ e^{-3} \end{pmatrix} = \frac{1+e^{-4}}{1+e^{-2}}$$

Lösung: $f(x) = \frac{1+e^{-4}}{1+e^{-2}} e^{(x-1)^2}$



5 Erweiterung und Abbildungen auf neuronale Netze

5.1 Erweiterung auf mehrwertige Funktionen

Bis jetzt wurden Funktionen betrachtet, deren Ausgabevektor eindimensional ist:
Ist jedoch der Ausgabevektor mehrdimensional, $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, kann das Problem auf den eindimensionalen Fall reduziert werden:
Betrachtet wird statt der Funktion $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ eine Funktionenschar mit Funktionen der Form:
$f : \mathbb{R}^n \rightarrow \mathbb{R} : y_l = f_l(X) = \sum_j c_{jl} h_j(\|X - W_j\|)$.
Für jede Komponente des Ausgabevektors steht ein Neuron in der Ausgabeschicht.
Die Zentren und die Zentrumsfunktionen sind für jede Funktion gleich. Die Funktionen unterscheiden sich nur in der Gewichtung zwischen den Neuronen in der verdeckten Schicht und der Ausgabeschicht.

5.2 Iteratives Nachtraining der RBF-Netze

Folge Gründe sprechen für ein iteratives Nachtraining:

- Optimierung der Stützstellen-Vektoren
- Bei großen Gewichten treten bei der numerischen Berechnung von Inversen und Pseudoinversen große Abweichungen auf
  - entweder normales iteratives Nachtraining oder
  - Verzicht auf die numerische Berechnung, Start mit kleinen Werten und iterative Berechnung

Das Ziel des Nachtrainings ist es, den Gesamtfehler
$E = \frac{1}{2} \sum_{l=1}^{m} \sum_{i=1}^{N} (y_{il} - o_l(X_i))^2$
zu minimieren. E summiert die Fehler über alle Ausgabeneuronen und über alle Trainingsmuster auf.

Die Minimierung erfolgt über den *Backpropagation-Algorithmus*:
Jede Stützstelle W_j und jeder Gewichtskoeffizient c_{jl} wird um einen durch die Lernrate η festgelegten Bruchteil des negativen Gradienten der Fehlerfunktion geändert:

- $\Delta W_j = -\eta_1 \frac{\delta E}{\delta W_j}$
- $\Delta c_{jl} = -\eta_2 \frac{\delta E}{\delta c_{jl}}$

Hierbei wird das *batch-Verfahren* angewendet:
Es werden erst die Fehler über alle Trainingsmuster berechnet und dann die Gewichte modifiziert. Im Gegensatz dazu steht das *online-Verfahren*, das nach jedem Trainingsmuster die Gewichte verändert. Das hier verwendete Verfahren stellt sicher, dass der Gesamtfehler stetig minimiert wird.

Durch den Momentum-Term kann das Verfahren noch verbessert werden:
Die Veränderung der Parameter ist nicht nur von den derzeitigen, sondern auch von den vorherigen Gradienten abhängig. Den Grad der Abhängigkeit bestimmt der Parameter μ. Haben die zwei Gradienten das gleiche Vorzeichen, so verläuft die Gewichtsmodifikation schneller. Sind die Vorzeichen verschieden, verläuft sie langsamer.
$\Delta a(t+1) = -\eta \frac{\delta E}{\delta a} + \mu \Delta a(t)$
Damit wird die Modifikation in flachen Plateaus vorangetrieben und in der Nähe eines Extremums verlangsamt.

6 Wahl der Zentren und Radien in RBF-Netzen

Um bessere Ergebnisse zu erzielen, ist eine optimale Wahl von Zentren und Radien wichtig. Eine Möglichkeit für die Auswahl ist das Lernverfahren mit harter Vektorquantisierung.

6.1 Wahl der Zentren und Radien

Folgender Algorithmus beschreibt die spezielle LVQ-Variante k-means. Im Allgemeinen wird der Vektor X jeweils dem Zentrum zugeordnet, für das die Zugehörigkeitswahrscheinlichkeit maximal ist.

Zuerst werden rein zufällig oder auf Grund von Vorkenntnissen K Eingabevektoren aus den N Trainingsmustern als Zentren $W_1, \cdots, W_k$ ausgewählt.
Danach werden die Eingabevektoren aus den Trainingsmustern nacheinenander mit den Zentrumsvektoren verglichen.

1. Für jeden Eingabevektor X wird derjenige Zentrumsvektor ausgesucht, der ihm am nächsten ist:
   $\|X - W_c\| = \min_j(\|X - W_j\|)$

2. Je nachdem, ob der Eingabevektor X zu der gleichen Klasse wie W_c gehört oder nicht, wird der Gewichtsvektor auf verschiedene Art und Weise verändert:

   $$W_c(t+1) = \begin{cases} W_c(t) + \beta_t(X(t) - W_c(t)) & \text{falls Klasse}(W_c) = \text{Klasse}(X) \\ W_c(t) - \beta_t(X(t) - W_c(t)) & \text{falls Klasse}(W_c) \neq \text{Klasse}(X) \end{cases}$$

   wobei $0 < \beta_t < 1$ ist.
   Die anderen Gewichtsvektoren bleiben unverändert.

3. Danach wird der Radius neu bestimmt. Dieser ist abhänig von der Standardabweichungen der Klasse. Sie wird von jeder einzelnen Klasse berechnet:

   $$\sigma_j = \sqrt{\frac{1}{\|Klasse(W_j)\|-1} \sum_{i=1}^{\|Klasse(W_j)\|} (X - W_j)^2},$$

   wobei $X \in$ Klasse(W_j)

   Der Mittelwert der Standardabweichungen aller Klassen bildet den neuen Radius:

   $$r = \frac{1}{k} \sum_{j=1}^{k} \sigma_j$$

Dieser Algorithmus wird solange wiederholt bis $|W_j(t+1) - W_j(t)| < \epsilon$, wobei ϵ ein konstanter, sehr kleiner Wert ist.

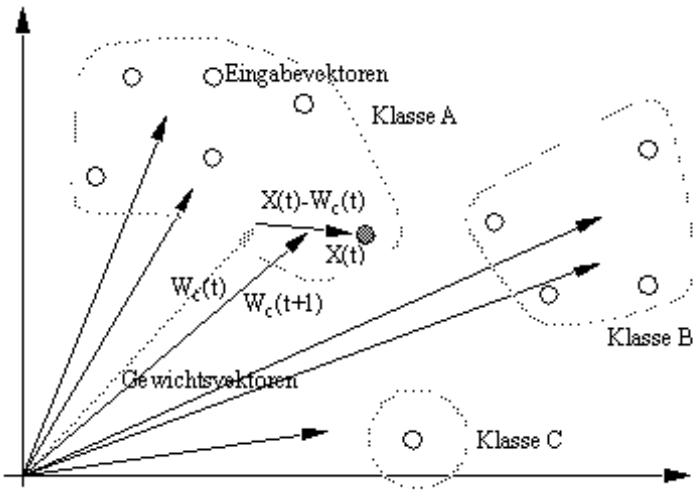


Abbildung 5: Beispiel des LVQ-Verfahrens

Unterschied zur weichen Vektorquantisierung:
Für jeden Eingabevektor X wird die Wahrscheinlichkeit bestimmt, dass er in die Klasse W_c gehört. Proportional zu seiner Wahrscheinlichkeit wird jeder Zentrumsvektor anhand obiger Formel neu berechnet.

7 Zusammenfassung

RBF-Netze sind *feed-forward-Netze* mit nur einer inneren Schicht. Bei relativ kleinen Netzen können die Gewichte direkt berechnet werden. Damit ist das Tainingsverfahren um einiges effizienter als das bei mehrstufigen Perzeptronen.

Die Aktivierungsfunktionen sind radialsymmetrisch. Damit eignen sich die Netze sehr gut für lokale Klassifizierungsprobleme.
Bei Punkten, die weiter von den Zentren der Aktivierungsfunktionen entfernt sind, geht die Ausgabe Netzes des gegen Null. Damit unterscheiden sie sich von den mehrstufigen Perzeptronen, die für solche Werte unvorhergesehene Ausgaben liefern.

Literatur

[Zell94] Zell, A. : Simulation neuronaler Netze. Oldenbourg 1994.