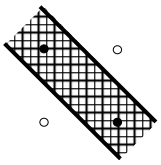
Demo mit MATLAB

(Demo von Theodoridis)

- C:\...\matlab\PR\startdemo
- Example 2 (XOR) mit (2-2-1 und 2-5-5-1)
- Example 3 mit dreischichtigem Netzwerk [5,5] (3000 Epochen, learning rate 0,7, momentum 0,5)

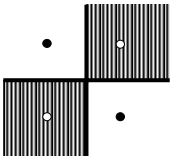
Start der Matlab-Demo
matlab-theodoridis.bat

Zwei Lösungen des XOR-Problems:



Konvexes Gebiet realisiert mit einem zweischichtigen Netz (2-2-1). Mögliche Fehlklassifikation bei Varianz:

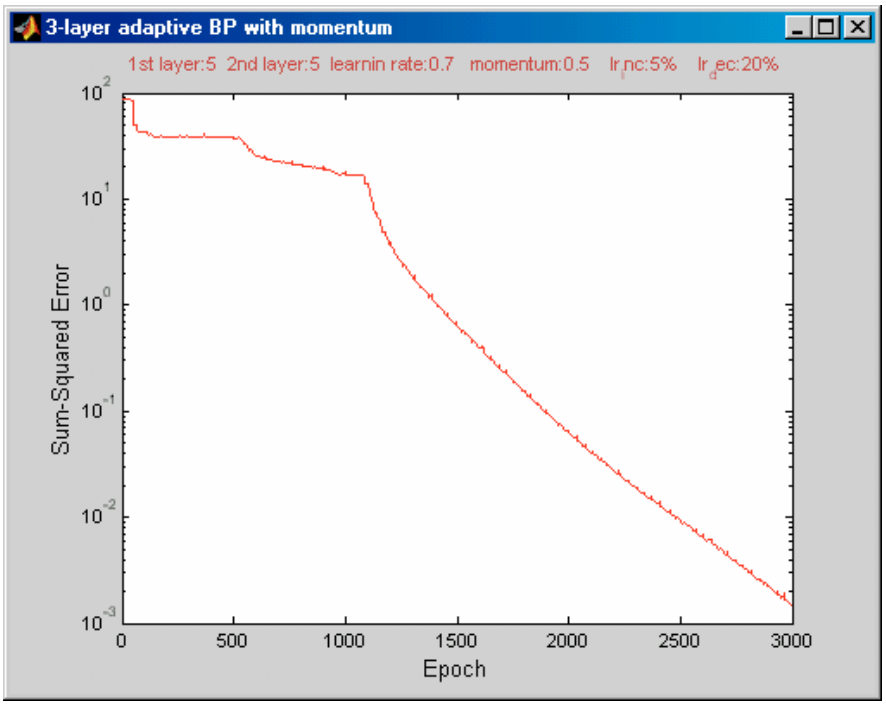
$$\|\mathbf{n}\| > \tfrac{1}{4}\sqrt{2} \approx 0{,}35$$

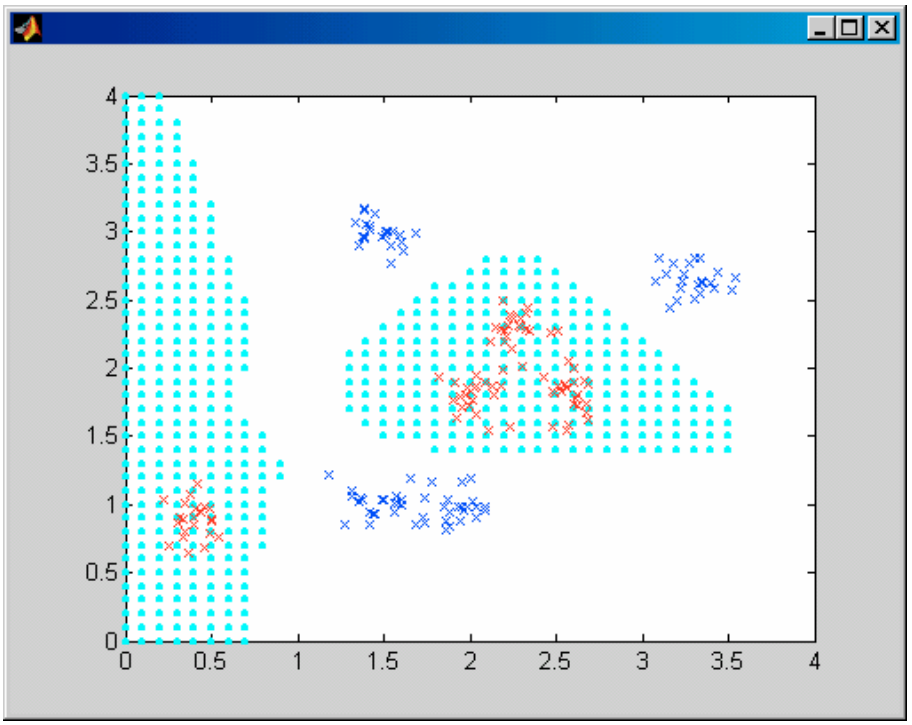


2-2-2-1 konvergiert leider nicht!

Vereinigung von 2 konvexen Gebieten realisiert mit einem dreischichtigen Netz (2-5-5-1). Mögliche Fehlklassifikation bei Varianz:

$$\|\mathbf{n}\| > 0{,}5$$



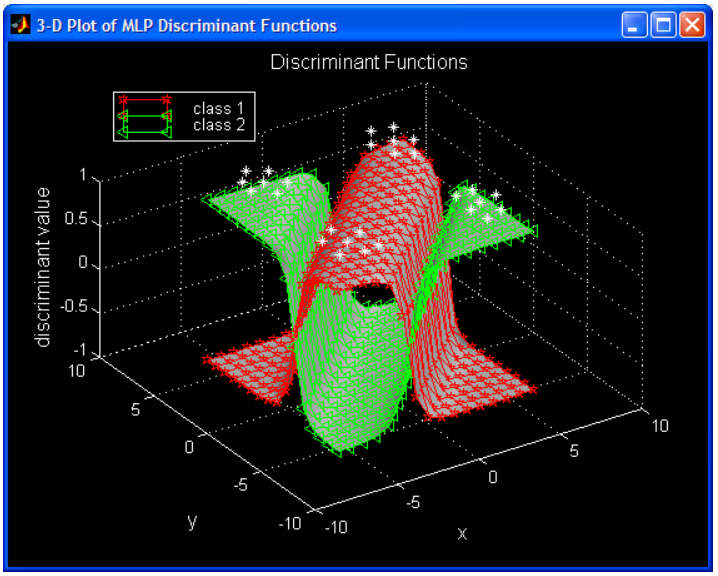


Demo mit MATLAB
(Klassifikationgui.m)

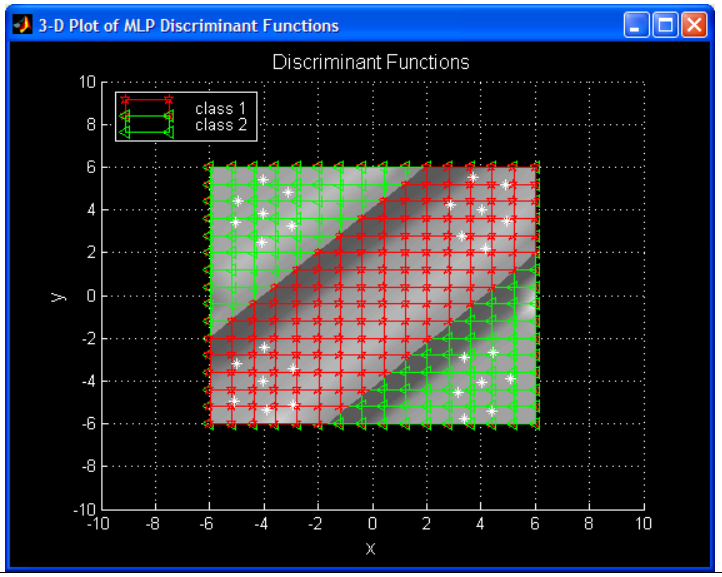
- Öffnen von Matlab
  - zuerst setmypath.m aufrufen, dann
  - C:\Home\ppt\Lehre\ME_2002\matlab\KlassifikatorEntwurf-WinXX\Klassifikationgui
  - Datensatz xor2.mat laden
  - Gewichtsmatrizen xor-trivial.mat laden
  - Gewichtsmatrizen xor-3.mat laden
- Zweiklassenproblem mit Bananenshape eingeben
  - Datensatz Samples/banana_test_samples.mat laden
  - Voreinstellungen aus Models/MLP_5_3_2_banana.mat laden

Start der Matlab-Demo
matlab-Klassifikation_gui.bat

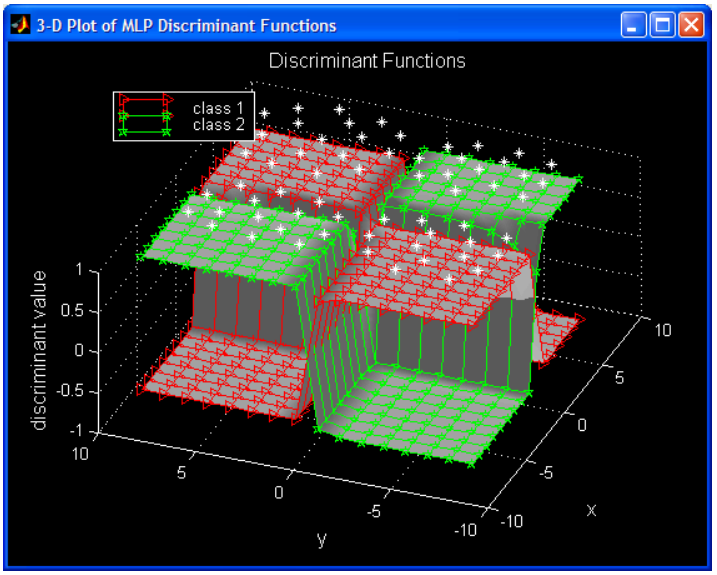
Lösung des XOR-Problems mit zweischichtigem NN



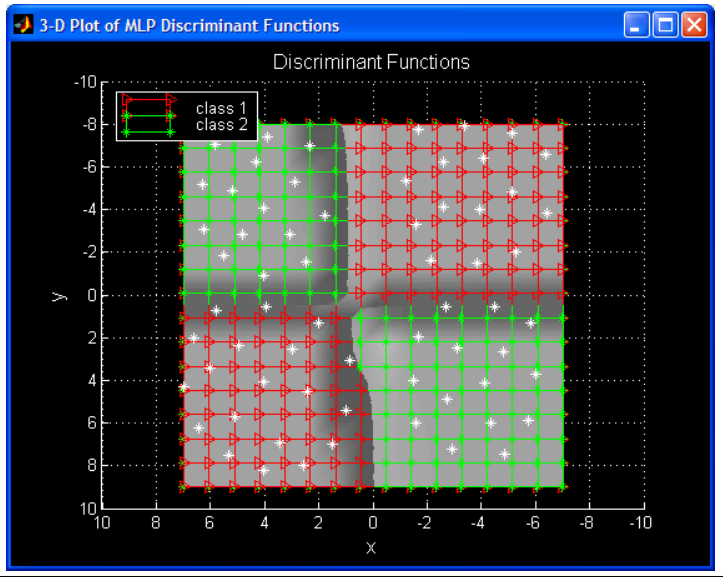
Lösung des XOR-Problems mit zweischichtigem NN



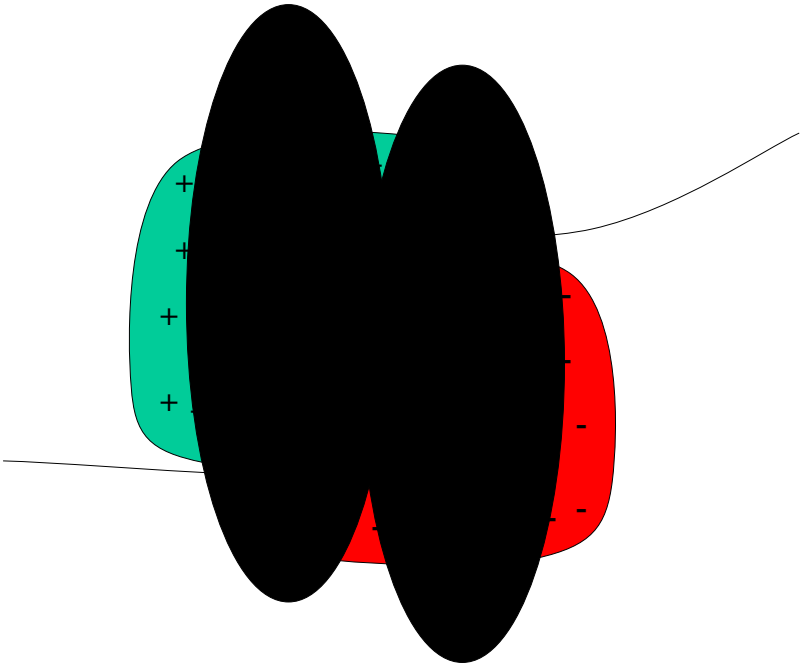
Lösung des XOR-Problems mit dreischichtigem NN

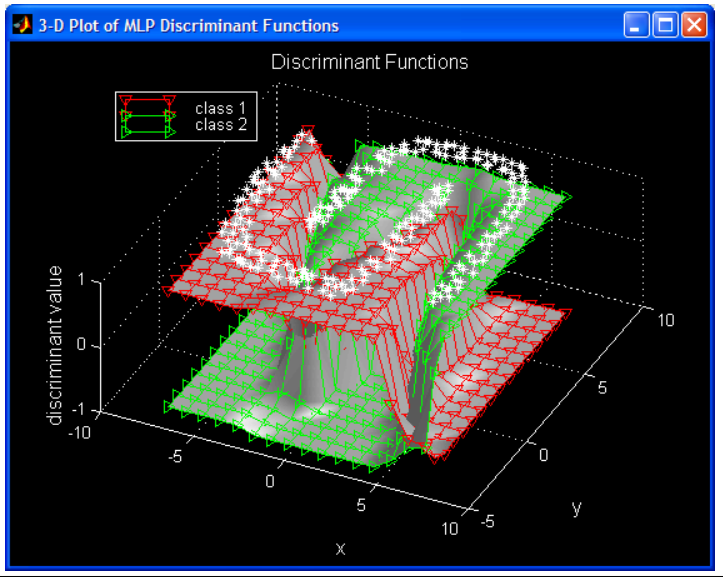


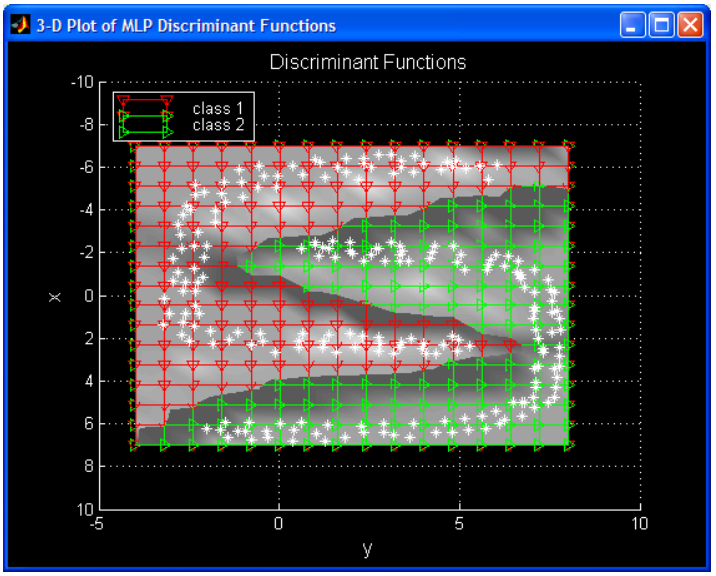
Lösung des XOR-Problems mit dreischichtigem NN



Durch Normalverteilungen schlecht darstellbare Klassen







Konvergenzverhalten von Backpropagation-Algorithmen

- Backpropagation mit gewöhnlichem Gradientenabstieg (steepest descent)

Start der Matlab-Demo
matlab-BP-gradient.bat

- Backpropagation mit konjugiertem Gradientenabstieg (conjugate gradient)

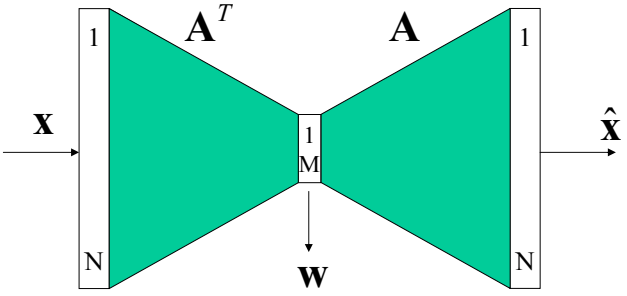
Start der Matlab-Demo
matlab-BP-CGgradient.bat

NN und deren Eigenschaften

- „Quick and Dirty“. Ein NN-Entwurf ist einfach zu realisieren und man erhält durchaus brauchbare Lösungen (eine suboptimale Lösung ist jedenfalls besser als irgendeine Lösung).
- Es können damit insbesondere auch sehr große Probleme angegangen werden.
- Alle Strategien benutzen jedoch nur „lokale“ Optimierungsfunktionen und erreichen in der Regel kein globales Optimum. Dies ist der gravierende Nachteil für den Einsatz Neuronaler Netze.

Verwendung eines NN zur iterativen Berechnung der Hauptkomponentenanalyse (KLT)

- Anstatt die KLT explizit über ein Eigenwertproblem zu lösen, kann auch ein NN zur iterativen Berechnung herangezogen werden.
- Man verwendet ein zweischichtiges Perceptron. Der Ausgang der verdeckten Schicht **w** ist der gesuchte Merkmalsvektor.



- Die erste Schicht berechnet den Merkmalsvektor zu:

$$\mathbf{w} = \mathbf{A}^T \mathbf{x}$$

- Dabei wird angenommen, dass eine *lineare Aktivierungsfunktion* zum Ansatz kommt. Die zweite Schicht realisiert die Rekonstruktion von $\mathbf{x}$ mit der transponierten Gewichtsmatrix der ersten Schicht

$$\hat{\mathbf{x}} = \mathbf{A}\mathbf{w}$$

- Das Optimierungsziel ist die Minimierung von:

$$J = E\left\{\left\|\hat{\mathbf{x}} - \mathbf{x}\right\|^2\right\} = E\left\{\left\|\mathbf{A}\mathbf{w} - \mathbf{x}\right\|^2\right\} = E\left\{\left\|\mathbf{A}\mathbf{A}^T\mathbf{x} - \mathbf{x}\right\|^2\right\}$$

- Die folgende Lernregel:

$$\mathbf{A} \leftarrow \mathbf{A} - \alpha(\mathbf{A}\mathbf{w} - \mathbf{x})\mathbf{w}^T = \mathbf{A} - \alpha\nabla\mathbf{J}$$

- führt zu einer Koeffizientenmatrix $\mathbf{A}$, welche als Produkt von zwei Matrizen geschrieben werden kann (ohne Beweis!):

$$\mathbf{A} = \mathbf{B}_M\mathbf{T}$$

- $\mathbf{B}_M$ ist eine $N \times M$-Matrix der M dominanten Eigenvektoren von $E\{\mathbf{x}\mathbf{x}^T\}$ und $\mathbf{T}$ eine orthonormale $M \times M$–Matrix, welche eine Rotation des Koordinatensystems bewirkt, innerhalb eines Raumes, welcher durch die M dominanten Eigenvektoren von $E\{\mathbf{x}\mathbf{x}^T\}$ aufgespannt wird.

- Die Lernstrategie beinhaltet keine Translation. D.h. sie berechnet die Eigenvektoren der Momentenmatrix $E\{\mathbf{x}\mathbf{x}^T\}$ und nicht der Kovarianzmatrix $\mathbf{K} = E\{(\mathbf{x}-\boldsymbol{\mu}_x)(\mathbf{x}-\boldsymbol{\mu}_x)^T\}$. Dies kann jedoch realisiert werden, indem man einen rekursiv geschätzten Erwartunghswert $\boldsymbol{\mu}_x = E\{\mathbf{x}\}$ subtrahiert, bevor man die rekursive Schätzung des Merkmalvektors beginnt. Den gleichen Effekt erzielt man dadurch, dass man einen erweiterten Beobachtungsvektor verwendet:

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \quad \text{anstatt} \quad \mathbf{x}$$