EFME 2010 LU Exercise 4

Report

Gruppe 15:

Fritz Daniel - 0507049 - 935

Hiller Elias - 0525787 - 935

Sonderegger Josef - 0501625 - 935

1. Perceptron

Um den Gewichtsvektor w mittels Perceptron zu berechnen wird der im Rahmen der Vorlesung vorgestellte Online Training Algorithmus angewendet.

```
    % Online Perceptron Training Algorithm #330 - Skriptum
    while wrong == true && cycle < maxEpoches

        cycle = cycle + 1;
        wrong = false;

        for i = 1:n
            if w'*(X(:,i)*t(i)) <= 0
                w = w + X(:,i)*t(i);
                wrong = true;
            end
        end
    end
```

Die Berechnung wird mit dem Aufruf `function [w] = perco(X,t,maxEpoches)` eingeleitet, wobei X den eigentlichen Daten und t dem Target entspricht. Die Anzahl der Epochen für den Algorithmus wird mit maxEpoches übergeben.

Das Ergebnis der Berechnung visualisieren wir anschließend in einem Plot, in den wir auch die decision boundary einzeichnen. Des weiteren wird zum einfachen Vergleich die Fehlerrate ermittelt, indem falsch klassifizierte Datensätze anhand des Targets ermittelt werden.

```
% identify wrong classified objects
    dif_anz=0;
    for i=1:n
       if sign(w'*X(:, i)) ~= t(i);
           dif_anz=dif_anz+1;
       end
    end
```

Das Perceptron wird nun auf die Probleme OR, AND und XOR angewendet (siehe Abbildung 1 bis 3). Mit dem von uns verwendeten einstufigen Perceptron kann das OR und AND Problem

gelöst werden. Das XOR-Problem ist nicht durch eine Hyperebene separierbar, weshalb diese Problem nicht zu lösen ist. XOR könnte allerdings mit einem Multilayerperceptron repräsentiert werden.

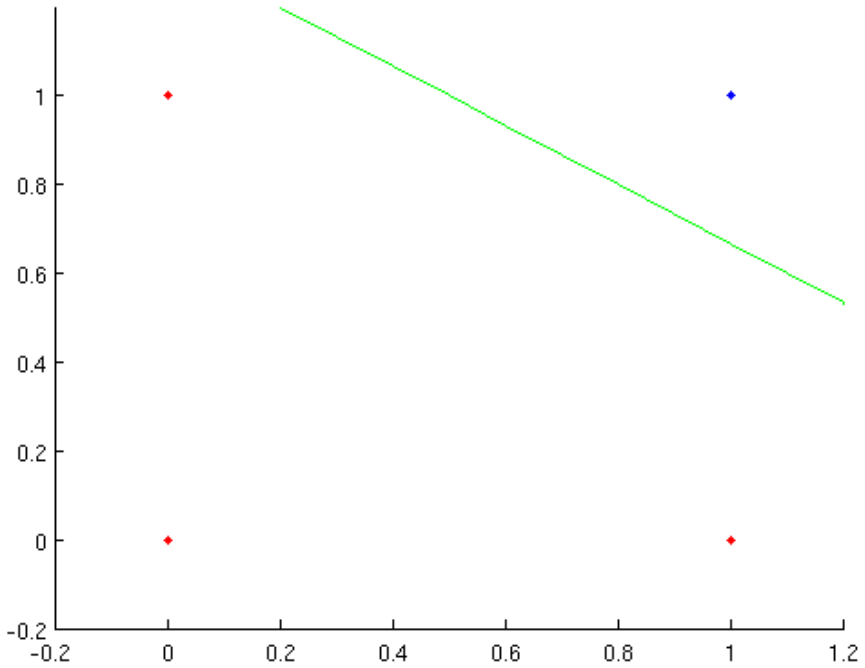

Abbildung 1: AND-Problem mit Entscheidungsgrenze

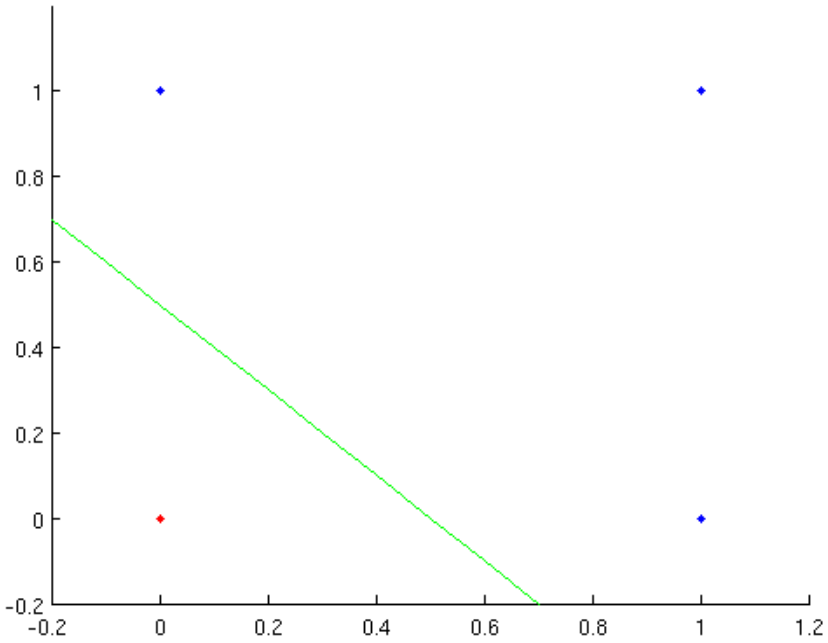

Abbildung 2: OR-Problem mit Entscheidungsgrenze

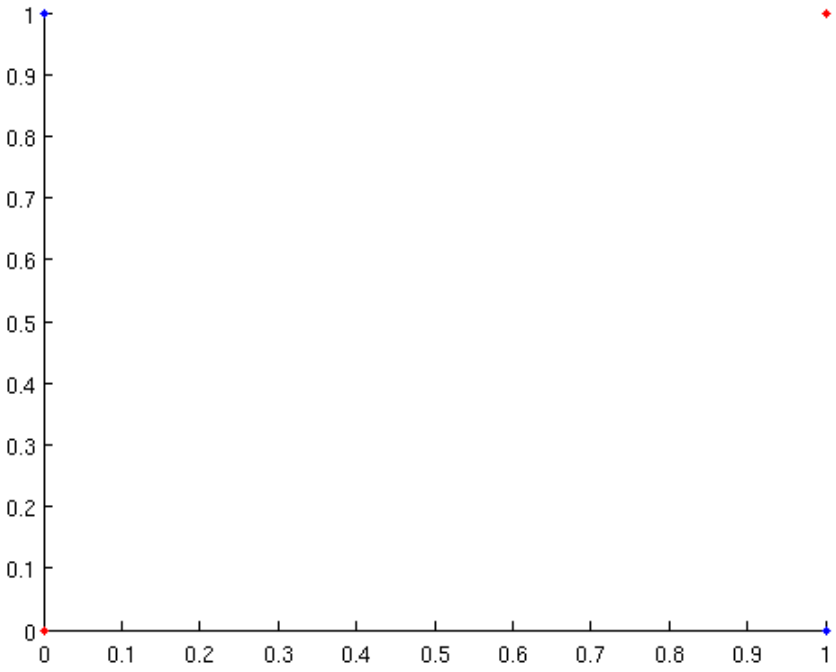

Abbildung 3: XOR-Problem ohne Entscheidungsgrenze

Um den Algorithmus zu testen, wurde er auf richtige Datensätze angewendet, welche im File perceptrondata gelistet sind. Dazu gab es zwei Files mit unterschiedlichen Targets (0,1),

dessen Ergebnisse es zu testen galt. Zu beachten ist hierbei, das die Daten zuallererst homogenisiert werden müssen und die Targets von 0,1 auf -1,1 umgewandelt werden. Ansonsten bleibt der Algorithmus derselbe.

Ergebnisse

Beim `Perceptrontarget1` sind die beiden Sets, wie in Abbildung 4 ersichtlich, sehr gut separierbar. Ab 10 Epochen ist bereits eine fehlerfreie Erkennung möglich und die beiden Datensets können vollständig voneinander unterschieden werden (Tabelle 1).

Epochen	Fehler	Erkennungsrate
1	1	99.761%
5	9	97.847%
10	0	100%
100	0	100%
1 000	0	100%
10 000	0	100%

Tabelle 1: Fehler, Erkennungsraten von `Perceptrontarget1` mit unterschiedlich vielen Epochen

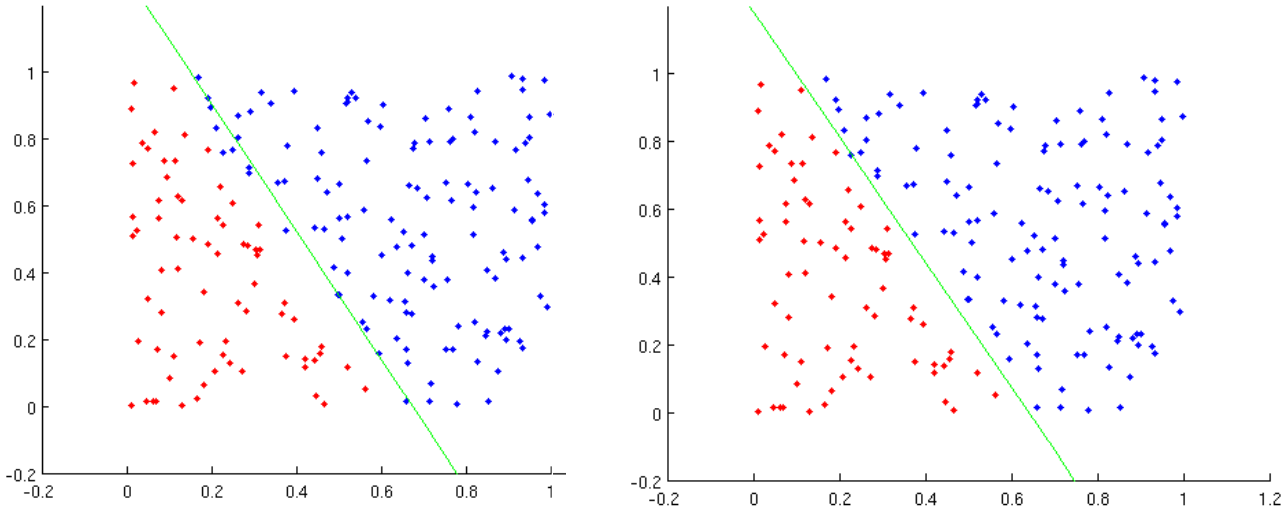

Abbildung 4: `Perceptronarget1` mit 5 Epochen (li) bzw 10 Epochen (re)

Beim `Perceptrontarget2` sind die Sets stärker vermischt, wodurch die Ergebnisse linear

nicht fehlerfrei separierbar sind. Auch mit einem enormen Aufwand von 1 000 000 Epochen bleiben 23 Fehler (siehe Tabelle 2).

Epochen	Fehler	Erkennungsrate
1	31	92.584%
10	55	86.842%
100	27	93.541%
1 000	55	86.842%
10 000	23	94.498%
100 000	28	93.301%
1 000 000	23	94.498%

Tabelle 2: Fehler, Erkennungsraten von `Perceptrontarget2` mit unterschiedlich vielen Epochen

Grafisch ist, wie in Abbildung 5 zu sehen ist, noch besser ersichtlich, wie die beiden Datensets sich überlappen und eine fehlerfreie Erkennung nicht möglich ist.

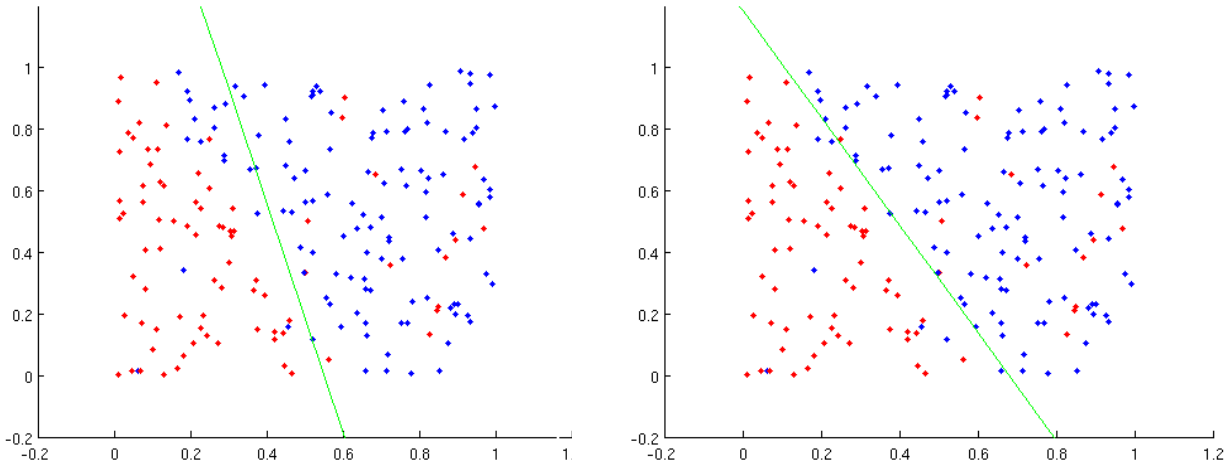

Abbildung 5: Perceptrontarget2 mit 10 Epochen (li) und 1 000 000 Epochen (re)