

# Einführung in die Neuroinformatik

Tim Luchterhand, Paul Nykiel (Gruppe P)

27. Juni 2018

## 1 Lernschritt im Perzeptron-Lernalgorithmus

(a)

$$\begin{aligned}w_1 \cdot x_1 + w_2 \cdot x_2 + w_0 &= 0 \\ \Leftrightarrow x_2 &= -\frac{w_1 \cdot x_1 + w_0}{w_2} \\ x_2 &= -\frac{1}{2}x_1 + \frac{1}{2}\end{aligned}$$



(b)

$$w^* = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}$$

$$x^* = \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$$

(c)  $w_1$  in rot,  $w_{-1}$  in blau



(d) Überprüfen ob bereits korrekt klassifiziert:

$$\begin{aligned} (w^*)^T \cdot x^* &= 2 + 4 - 1 = 5 \geq 0 \\ \Rightarrow x^* &\in \omega_1 \end{aligned}$$

Lernschritt durchführen:

$$\begin{aligned} \tilde{w}^* &= w^* - \eta \cdot x^* = \begin{pmatrix} -1 \\ 0 \\ -2 \end{pmatrix} \\ \Rightarrow \tilde{w} &= \begin{pmatrix} -1 \\ 0 \end{pmatrix} \\ \tilde{w}_0 &= -2 \end{aligned}$$

(e)  $w_1$  in rot,  $w_{-1}$  in blau



(f) Überprüfen ob bereits korrekt klassifiziert:

$$\begin{aligned}
 (\tilde{w}^*)^T \cdot x^* &= -2 + 0 - 2 = -4 \\
 &\Rightarrow x^* \in \omega_{-1}
 \end{aligned}$$

Kein Lernschritt ist notwendig  $\Rightarrow w$  wird nicht verändert

## 2 Perzeptron-Lernalgorithmus

Matlab script:

```

1 %% Initialization
2 data = [-3 1 -1;
3         -3 3 1;
4         -2 1 -1;
5         -2 4 1;
6         -1 3 1;
7         -1 4 1;
8         2 2 -1;
9         2 4 1;
10        3 2 -1;
11        4 1 -1;];
12
13
14 % Create all vectors
15 inputs = data(:,1:2);
16 inputsExtended = [inputs ones(size(inputs,1),1)];

```

```

17 classes = data(:,3);
18 w = [0,0,0];
19
20 % Extended weight vectors; each iteration adds one more row
21 % Since we don't know the exact number of rows in advance, we
    preallocate the matrix with a maximum size and crop the
    result in the end
22 maxVectors = 100;
23 vectorDimension = 3;
24 wExtendedMat = zeros(maxVectors, vectorDimension);
25 L = size(data, 1);
26 numberOfOptimizations = 1;
27 wExtendedMat(numberOfOptimizations,:) = w;
28
29 changesInLastIteration = 1;
30 while changesInLastIteration > 0
31     changesInLastIteration = 0;
32     for c = 1:size(inputs,1)
33         currentInput = inputsExtended(c,:);
34         desiredClass = classes(c,:);
35         calculatedOutput = w * transpose(currentInput);
36
37         % Wrong class
38         if calculatedOutput <= 0 && desiredClass > 0
39             w += currentInput;
40         elseif calculatedOutput >= 0 && desiredClass < 0
41             w -= currentInput;
42         end
43
44         % Plot the updated weights
45         if calculatedOutput * desiredClass <= 0
46             changesInLastIteration += 1;
47             numberOfOptimizations += 1;
48             wExtendedMat(numberOfOptimizations,:) = w;
49
50             x = y = 0;
51             if w(2) == 0
52                 y = -6:0.1:6;
53                 x = -w(3)/w(1) * ones(size(y));
54             else
55                 x = -6:0.1:6;
56                 y = -(w(1)*x + w(3))/w(2);
57             end
58             clf();

```

```

59     p = plot(x,y);
60     set(p, "linewidth", 1.5, "color", "green");
61     hold on;
62     scatter(data(:,1), data(:,2), [], data(:,3), 'filled',
63             );
64     xlabel("x_1");
65     ylabel("x_2");
66     title("Punkte in der Ebene und Separierungslinie");
67     axis([-6 6 -6 6]);
68     grid on;
69
70     if numberOfOptimizations <= 2
71         print("initial", "-depsec", "-color");
72         print(gcf, "initial.epsec");
73     else
74         print("separated", "-depsec", "-color");
75         print(gcf, "separated.epsec");
76     end
77
78     while waitforbuttonpress ~ = 1
79     end
80 end
81
82 end
83 end
84
85 % Remove unused weight vector entries
86 wExtendedMat = wExtendedMat(1:numberOfOptimizations, :);
87 L = 0;

```