## Einführung in die Neuroinformatik

Tim Luchterhand, Paul Nykiel (Gruppe P)

4. Juni 2018

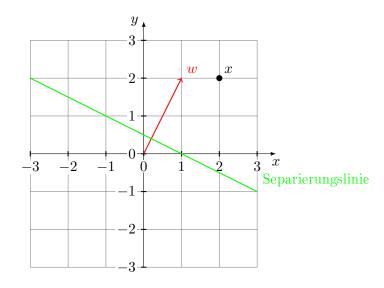
## 1 Lernschritt im Perzeptron-Lernalgorithmus

(a)

$$w_{1} \cdot x_{1} + w_{2} \cdot x_{2} + w_{0} = 0$$

$$\Leftrightarrow x_{2} = -\frac{w_{1} \cdot x_{1} + w_{0}}{w_{2}}$$

$$x_{2} = -\frac{1}{2}x_{1} + \frac{1}{2}$$



(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:

$$(w^*)^T \cdot x^* = 2 + 4 - 1 = 5 \ge 0$$
  
$$\Rightarrow x^* \in \omega_1$$

Lernschritt durchführen:

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

$$\Rightarrow \tilde{w} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

$$\tilde{w}_0 = -2$$

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



(f) Überprüfen ob bereits korrekt klassifiziert:

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

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

## 2 Perzeptron-Lernalgorithmus

Matlab script:

```
% Initialization
   data = [-3 \ 1 \ -1;
            -3 \ 3 \ 1;
            -2 \ 1 \ -1;
            -2\ 4\ 1;
            -1 \ 3 \ 1;
            -1 \ 4 \ 1;
             2 \ 2 \ -1;
              2 4 1;
              3 \ 2 \ -1;
10
              4 \ 1 \ -1;];
11
12
  % Create all vectors
  inputs = data(:,1:2);
   inputsExtended = [inputs ones(size(inputs,1),1)];
```

```
classes = data(:,3);
  w = [0, 0, 0];
19
  % Extended weight vectors; each iteration adds one more row
  % 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
  maxVectors = 100;
  vectorDimension = 3;
  wExtendedMat = zeros (max Vectors, vector Dimension);
  L = size(data, 1);
25
  numberOfOptimizations = 1;
  wExtendedMat(numberOfOptimizations,:) = w;
28
  changesInLastIteration = 1;
29
  while changesInLastIteration > 0
30
       changesInLastIteration = 0;
31
       for c = 1: size (inputs, 1)
32
           currentInput = inputsExtended(c,:);
           desiredClass = classes(c,:);
34
           calculatedOutput = w * transpose(currentInput);
35
36
           % Wrong class
37
           if calculatedOutput <= 0 && desiredClass > 0
38
               w += currentInput;
39
           elseif calculatedOutput >= 0 && desiredClass < 0
               w -= currentInput;
41
           end
42
43
           % Plot the updated weights
44
           if calculatedOutput * desiredClass <= 0
                changesInLastIteration += 1;
46
               numberOfOptimizations += 1;
47
               wExtendedMat(numberOfOptimizations,:) = w;
48
49
               x = y = 0;
50
               if w(2) == 0
51
                    y = -6:0.1:6;
                    x = -w(3)/w(1) * ones(size(y));
53
                else
54
                    x = -6:0.1:6;
55
                    y = -(w(1)*x + w(3))/w(2);
56
               end
               clf();
58
```

```
p = plot(x, y);
59
               set(p, "linewidth", 1.5, "color", "green");
60
               hold on;
61
               62
                  );
               xlabel("x 1");
63
               ylabel("x_2");
64
               title ("Punkte in der Ebene und Separierungslinie");
65
               axis([-6 \ 6 \ -6 \ 6]);
               grid on;
67
68
               if numberOfOptimizations <= 2
69
                   print("initial", "-depsc", "-color");
70
                   print(gcf, "initial.epsc");
               else
72
                   print("separated", "-depsc", "-color");
73
                   print(gcf , "separated.epsc");
74
75
               end
77
               while waitforbuttonpress \tilde{}=1
               end
79
           end
80
81
      end
82
  end
83
84
  % Remove unused weight vector entries
  wExtendedMat = wExtendedMat(1:numberOfOptimizations, :);
^{87} L = 0;
```