Einführung in die Neuroinformatik

Tim Luchterhand, Paul Nykiel (Gruppe P)

10. Juli 2018

1 Kohonen's selbstorganisierende Karte

1.1

(a)

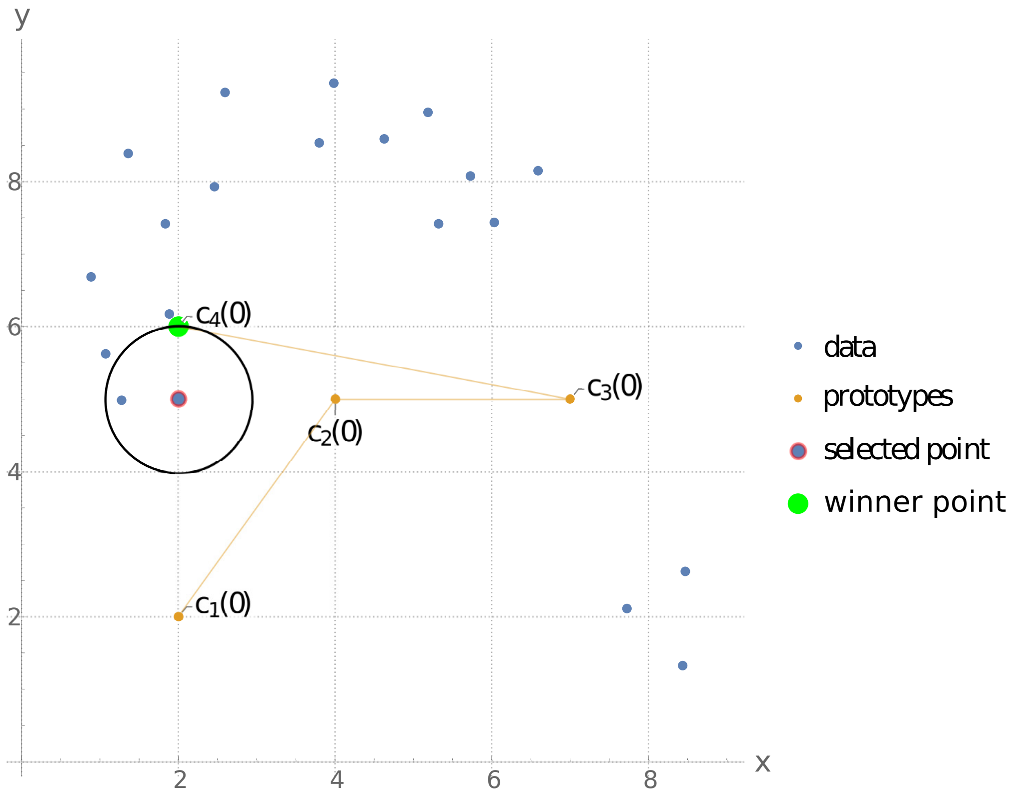

Abbildung 1: Graphische Bestimmung des Gewinners

(b)

$$
\begin{aligned}
c_j(t+1) &= c_j(t) + \eta(t) \cdot \mathcal{N}_t(g_j, g_{j^*}) \cdot (x - c_j(t)) \\
\eta(0) &= \eta_{\text{start}} = 1 \\
c_1(1) &= \begin{pmatrix} 2 \\ 2 \end{pmatrix} + \eta(0) \cdot \mathcal{N}_0(1,4) \cdot \left(\begin{pmatrix} 2 \\ 5 \end{pmatrix} - \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right) \\
&= \begin{pmatrix} 2 \\ 2.0003 \end{pmatrix} \\
c_2(1) &= \begin{pmatrix} 4 \\ 5 \end{pmatrix} + \eta(0) \cdot \mathcal{N}_0(2,4) \cdot \left(\begin{pmatrix} 2 \\ 5 \end{pmatrix} - \begin{pmatrix} 4 \\ 5 \end{pmatrix} \right) \\
&= \begin{pmatrix} 3.9662 \\ 5 \end{pmatrix} \\
c_3(1) &= \begin{pmatrix} 7 \\ 5 \end{pmatrix} + \eta(0) \cdot \mathcal{N}_0(3,4) \cdot \left(\begin{pmatrix} 2 \\ 5 \end{pmatrix} - \begin{pmatrix} 7 \\ 5 \end{pmatrix} \right) \\
&= \begin{pmatrix} 5.198 \\ 5 \end{pmatrix} \\
c_4(1) &= \begin{pmatrix} 2 \\ 6 \end{pmatrix} + \eta(0) \cdot \mathcal{N}_0(4,4) \cdot \left(\begin{pmatrix} 2 \\ 5 \end{pmatrix} - \begin{pmatrix} 2 \\ 6 \end{pmatrix} \right) \\
&= \begin{pmatrix} 2 \\ 5 \end{pmatrix}
\end{aligned}
$$

1.2

Die Nachbarschaftserhaltung ist eine allgemeine Eigenschaft des Algorithmus, da letzendlich versucht wird, ein Gitter wie aus $\{g_1, \ldots, g_n\}$ geschickt in die Datenpunkte zu legen.

1.3

Zu Beginn des Lernvorgangs liegen die Prototypen größtenteils ungünstig in den Datenpunkte verteilt, es müssen größere Anpassungen vorgenommen werden. Deshalb ist die generelle Lernrate η anfangs hoch. Gleiches gilt für den Nachbarschaftsweitenparameter σ. Beide Parameter nehmen mit der Zeit ab, damit gegen Ende nur noch kleine Anpassungen vorgenommen werden, um Konvergenz zu ermöglichen. Der Unterschied zwischen beiden Parametern ist, dass η den gesamten Lernvorgang auf Dauer abklingen lässt, während σ dafür sorgt, dass für große t die Anpassung für alle Prototypen außer dem Winner-Prototypen vernachlässigbar klein ist.

1.4

a)

$$\|c_1 - c_2\| = \|\begin{pmatrix}8.17\\2.06\end{pmatrix} - \begin{pmatrix}5.48\\8.14\end{pmatrix}\| = 6.65$$

b) Die Kantengewichte im Gitter sind ein ungefähres Maß für die Abstände der Datencluster, also ein Maß wie sehr sich die identifiezierten Klassen unterscheiden.

2 Kohonen-Karten als Visualisierungsinstrument

```
%% SOM network
rng(1337, 'combRecursive');
% TODO: train the network and process the result
load('titanic.mat');
maxVals = max(data);
minVals = min(data);

titanicNormalized = (data - minVals)./(maxVals - minVals);

%net = selforgmap([30 30], 400, 3, 'gridtop');

% Before you train the network, initialize the weights with the
    provided initialization data
%net = configure(net, data');
load('weights.mat');
%net.IW{1} = initWeights;

%net = train(net,transpose(titanicNormalized));

prototypes = net.IW{1} .* (maxVals - minVals) + minVals;
maps = permute(reshape(prototypes, [30 30 7]), [2,1,3]);

out = net(transpose(titanicNormalized));
count = sum(transpose(out));
hits = transpose(reshape(count, [30 30]));

mapSurvived = round(maps(:,:,1));
mapSurvived(find(mapSurvived < 0)) = 0;
mapSurvived(find(mapSurvived > 1)) = 1;


```

```matlab
%% Plot some features (two examples are shown)
for feature=1:length(featureNames)
    figure;
    [ax1, ax2] = mapPlot(maps(:, :, feature), hits, mapSurvived);

    title(ax1, featureNames(feature));
    colorbar(ax2, 'Position', [0.88 0.11 0.0275 0.815]);
    print(featureNames(feature) + ".eps", "-depsc");
end



%%
function [ax1, ax2] = mapPlot(map, hits, mapSurvived)
%mapPlot creates the visualization for one map dimension
%   Arguments:
%       - map: prototype data for one map dimension, i.e. maps(:, :, i). Note that you need to round and clip the values yourself (if required)
%       - hits: matrix with the same size as one map dimension. Gives for each prototype the number of data points which are assigned to it
%       - mapSurvived: first map dimension used as background colour
%
%   Returns:
%       - ax1: Matlab axis object used for the background colouring (survived information). Use this axis to set e.g. the title
%       - ax2: Matlab axis object used to draw the points on. Use this axis to set the colorbar
%

    % Axes combination based on : https://de.mathworks.com/matlabcentral/answers/194554-how-can-i-use-and-display-two-different-colormaps-on-the-same-figure

    % Plot the survived area in the background
    ax1 = survivedPlot(mapSurvived);

    % Plot the current map
    ax2 = axes;

```

```
    map = double(map);
    mapValues = map(:);   % Scatter expects a list of points
    mapDistinct = unique(mapValues)';

    if all(all(map == floor(map)))
        % For integer values, use a color for every possible value in the range
        colors = winter(max(mapDistinct) - min(mapDistinct) + 1);
    else
        % For floating values, use a fixed number of colors
        colors = winter(256);
    end

    % Map each value to its corresponding color
    mapValues = (mapValues - min(mapValues)) / (max(mapValues) - min(mapValues));   % Scale to [0;1]
    mapValues = mapValues * (size(colors, 1) - 1) + 1;                              % Scale to available color range (e.g. [0;1] -> [0;255] -> [1;256])
    mapValues = round(mapValues);                                                    % Make sure we can use the map values as indices
    colorVec = colors(mapValues, :);                                                 % Color value for each map value

    % Plot the map as circles scaled by the number of hits
    [X, Y] = meshgrid(1:size(map, 1), 1:size(map, 2));
    hits(hits > 0) = hits(hits > 0) + 1.5;  % Set minimum size of points (zero-hits are not displayed)
    scatter(ax2, X(:)+0.5, Y(:)+0.5, (hits(:)+0.00001)*15, colorVec, 'filled');
    colormap(ax2, colors);
    axis square;
    xlim([1 size(map, 1)]);
    ylim([1 size(map, 2)]);

    % Set the color range to the data range
    range = [min(mapDistinct) max(mapDistinct)];
    caxis(ax2, range);

    % Combine both plots
    linkaxes([ax1, ax2]);
```

```

    % Hide the top axis
    ax2.Visible = 'off';
    ax2.XTick = [];
    ax2.YTick = [];
end

function [ax] = survivedPlot(map)
    % Based on: https://stackoverflow.com/questions/3280705/how-can-i-display-a-2d-binary-matrix-as-a-black-white-plot
    [rows, cols] = size(map);
    ax = axes;
    imagesc(ax, (1:cols)+0.5, (1:rows)+0.5, map);
    xlabel('first map dimension');
    ylabel('second map dimension');
    impixelinfo;
    axis square;
    axis xy

    % Color the two areas differently
    colorSurvived = [0.8 0.8 0.8];
    colorNotSurvived = [1 1 1];
    colormap(ax, [colorSurvived; colorNotSurvived]);

    % Manually specify the tick labels (in steps of 5)
    xTicks = 1:cols;
    xTicks(mod(xTicks, 5) ~= 0) = NaN;
    xTicks = replace(cellstr(num2str(xTicks')), 'NaN', '');

    yTicks = 1:rows;
    yTicks(mod(yTicks, 5) ~= 0) = NaN;
    yTicks = replace(cellstr(num2str(yTicks')), 'NaN', '');

    % A grid line is used at every position so that each matrix value gets its own rectangle
    set(gca, 'XLim', [1 cols+1], 'YLim', [1 rows+1], ...
        'GridLineStyle', '-', 'GridColor', 'black', 'GridAlpha', 1, ...
        'XGrid', 'on', 'YGrid', 'on', 'XTick', 1:(cols+1), 'YTick', 1:(rows+1), ...
        'XTickLabel', xTicks, 'YTickLabel', yTicks);
end
```

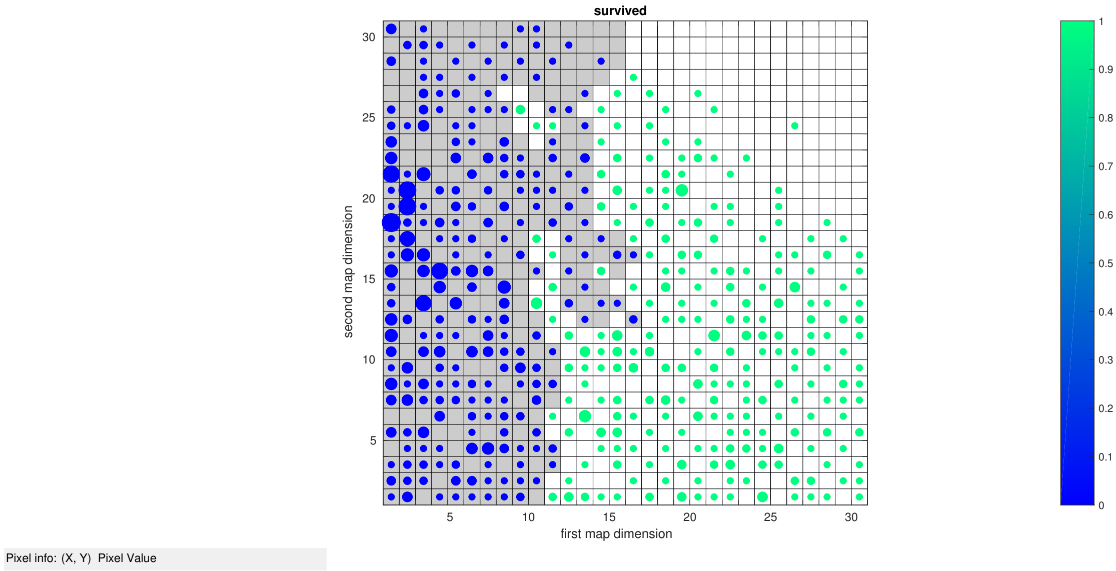

Abbildung 2: Ticketklasse

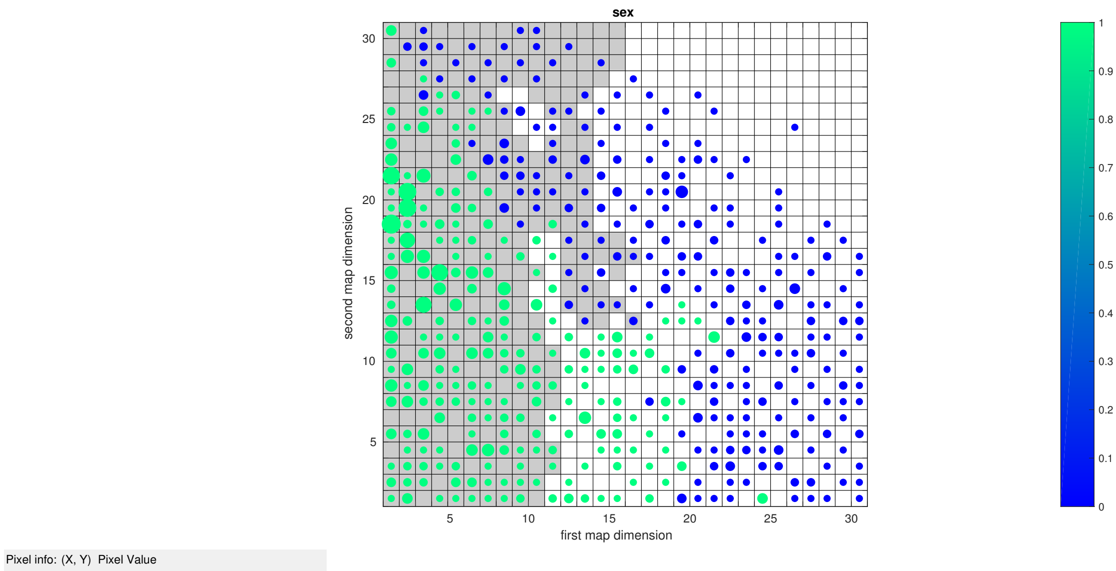

Abbildung 3: Geschlecht

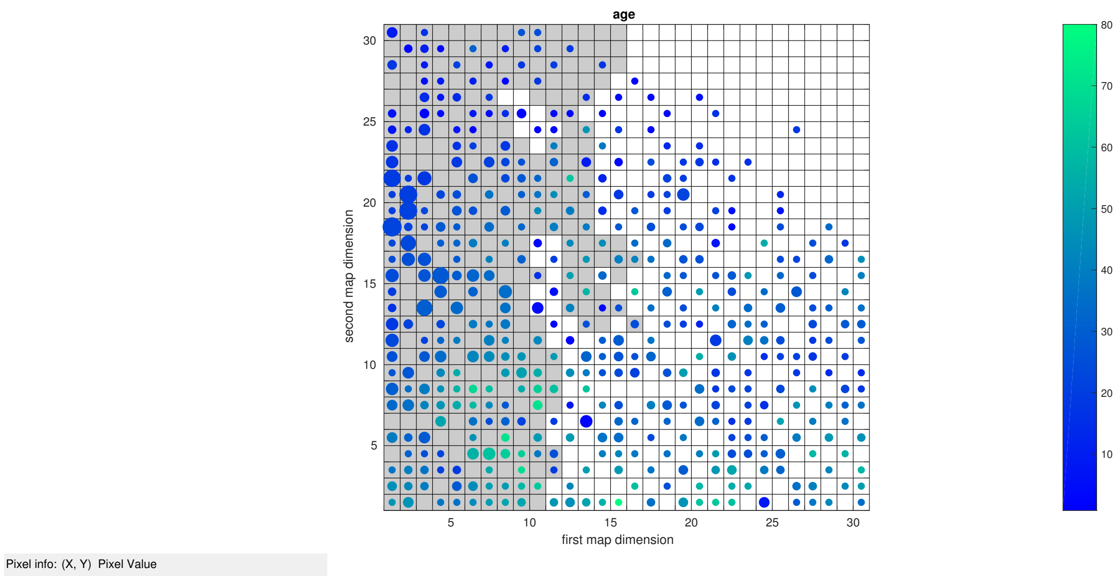

Abbildung 4: Alter