Open your mind. LUT.
Lappeenranta University of Technology
LUT Machine Vision and Pattern Recognition 2015-10-27

BM40A0700 Pattern Recognition

Lasse Lensu

Exercise 7: Linear classification

1. Perceptron (2 points): Construct a Matlab function

   ```
   w = percep(class, data)
   ```

   that performs the perceptron algorithm for two-dimensional data. The function should return the weight vector of the discriminant. Use the batch method for the update rule. The update should happen in real time on the screen so that the line that divides the two different datasets will be updated and drawn constantly as the algorithm progresses.

   Three data sets are given for the experiments. Which of the data sets can be classified using the perceptron? You can initialize **w** as a vector of random values (e.g., `w = rand(size(data, 1), 1);`). Also it is recommended that the learning rate ρ follows $\rho = 0.1/t$ where t is the current iteration round.

   Notes: Remember that you need to use the extended weight and training data vectors. The algorithm should be stopped when none of the samples are incorrectly classified.

   The code for plotting the data and the dividing line is as follows:

   ```
   % Initialization
   xmax = max(data(1, :)) + 1;
   xmin = min(data(1, :)) - 1;
   ymax = max(data(2, :)) + 1;
   ymin = min(data(2, :)) - 1;

   ...

   % Plot data and discriminant
   hold off;
   plot(data(1, class==1), data(2, class==1), 'bx', ...
     -data(1, class==2), -data(2, class==2), 'ro');
   axis([xmin xmax ymin ymax]);
   hold on;
   line([xmin xmax], [-(w(1) * xmin + w(3)) / w(2), ...
     -(w(1) * xmax + w(3)) / w(2)]);
   pause(0.1);
   ```

   Notes: In the code for plotting, the data for class two has '-' -sign before it. This is done because the data has been normalized. The program can made to run faster by plotting once every 50 iterations or so.

   As the end condition, you can use that either all the samples are correctly classified or that an iteration limit (e.g. 10000 iterations) is reached.

   Additional files: `data1.mat, data2.mat, data3.mat`.

PO BOX 20, FI-53851 LAPPEENRANTA, FINLAND
E-mail: lasse.lensu@lut.fi

2. Least mean square (LMS) linear classifier (1 point): Construct a Matlab function

   ```
   w = lms(class, data);
   ```

   that produces the linear discriminant according to the sum of squared error function. You can choose to use either the analytic solution or an iterative one (gradient descent). Plot the data and the discriminant. Three data sets are provided for the experiments.

   Hints: Use extended data vectors. The analytic solution can be easier to implement compared to the iterative one because it does not need parameters to be tuned.

   Concerning the iterative methods, the iteration can stop when $\|w(t+1) - w(t)\| < e$ (in Matlab `norm(w ./ norm(w) - wold ./ norm(wold)) < e`) where e should be a small positive constant, for example, 10^{-6}. In this task, the learning rate ρ could be $\rho = 0.5/t$ in order to make the function to converge. To initialize the weight $\mathbf{w}$, the following can be tried: `w = (rand(3,1) - 0.5) / 10;`

   Additional files: `data1.mat, data2.mat, data3.mat`.