## 3 $M$-fold Cross-validation

Implement and perform $M$-fold cross validation for LVQ1 training, using the data set of assignment 1.
First, shuffle the two-class data set randomly once, then split the data into $M = 5$ disjoint subsets, each containing 20% of the data. Take care that the correct labels are still assigned to the data points.
In an individual training run, use 4 of the $M = 5$ subsets for training and 1 subset for validation. Perform LVQ1 training as in assignment 1 with $K$ prototypes per class for up to 100 *epochs*. Initialize each prototype in a randomly selected point of its class.
Determine the *training error* (performance on the 4/5 of data used for training) and the *validation error* (performance on 1/5 of the data) **at the end of each training process**. Repeat this for the 5 possible splits into training and validation set to obtain 5 individual results.
Compute the average training and validation errors (i.e. the fraction of misclassified data points) and their standard deviations over the $M = 5$ results. Plot the results as a function of $K$, the number of LVQ prototypes per class. [2]
You should hand in a structured report comprising:

- **(2 point)** A brief introduction and description of your experiments.
- **(4 points)** Figures displaying both the final training and validation error as a function of $K = 1, 2, \ldots 5$ as obtained in 5-fold cross-validation (averages and standard deviation, see above). Of course you may obtain and present results for even greater values of $K$.
- **(4 points)** A discussion of your results. What is your conclusion with respect to the choice of the number $K$ of prototypes?

### Bonus (suggestions)

1 point max. in total:

- Consider also 10-fold cross validation $(M = 10)$ and compare with the results for $M = 5$.
- Repeat the cross-validation experiments for $K = 1$, now for different constant learning rates $\eta$. Can you determine a *best* value?
- Perform $M$-fold cross validation for the regression problem (and data set) of assignment 2.

[2] A useful matlab command for displaying the standard deviations is `errorbar(....)`.