Lab assignment 2 Radial basis functions, competitive learning and self-organisation

Alice Karnsund, Elin Samuelsson & Irene Natale

KTH The Royal Institute of Technology

September 20, 2018

Introduction

- know how to build the structure and perform training of an RBF network for either classification or regression purposes
- be able to comparatively analyse different methods for initializing the structure and learning the weights in an RBF network
- know the concept of vector quantisation and learn how to use it in NN context
- be able to recognize and implement different components in the SOM algorithm
- be able to discuss the role of the neighbourhood and analyse its effect on the self-organisation in SOMs
- know how SOM-networks can be used to fold high-dimensional spaces and cluster data

Weights matrix - Existence of solution

f is the target function, x_k the kth pattern, we want to solve the following system of equations

$$\Phi_1(x_1)w_1 + \Phi_2(x_1)w_2 + ... + \Phi_n(x_1)w_n = f(x_1)$$

$$\Phi_1(x_2)w_1 + \Phi_2(x_2)w_2 + ... + \Phi_n(x_2)w_n = f(x_2)$$

$$\Phi_1(x_N)w_1 + \Phi_2(x_N)w_2 + ... + \Phi_n(x_N)w_n = f(x_N)$$

1. What is the lower bound for the number of training samples? N = n
2. What happens with the error in N=n? Why? e=0, $f(\hat{x}_k) = f(x_k)$, the target f is perfectly recontructed

1. Under what conditions, if any, does the system have a solution in this case (N=n)? rank(columns) = rank(rows)
2. During training we used error measure defined over training examples. Is it good? It is better to measure it on the test dataset. We could train the network too much, overfitting, low error in train samples but higher in test samples.

PART 1: Batch mode training using least squares - supervised learning of network weights

- We used the absolute residual error throughout section 3.1 and 3.2.
- n denotes number of RBF nodes and σ their standard deviation.
- Plots with errorbars have been generated by either 10, 50 or 100 training sessions.

Function approximation with RBF networks

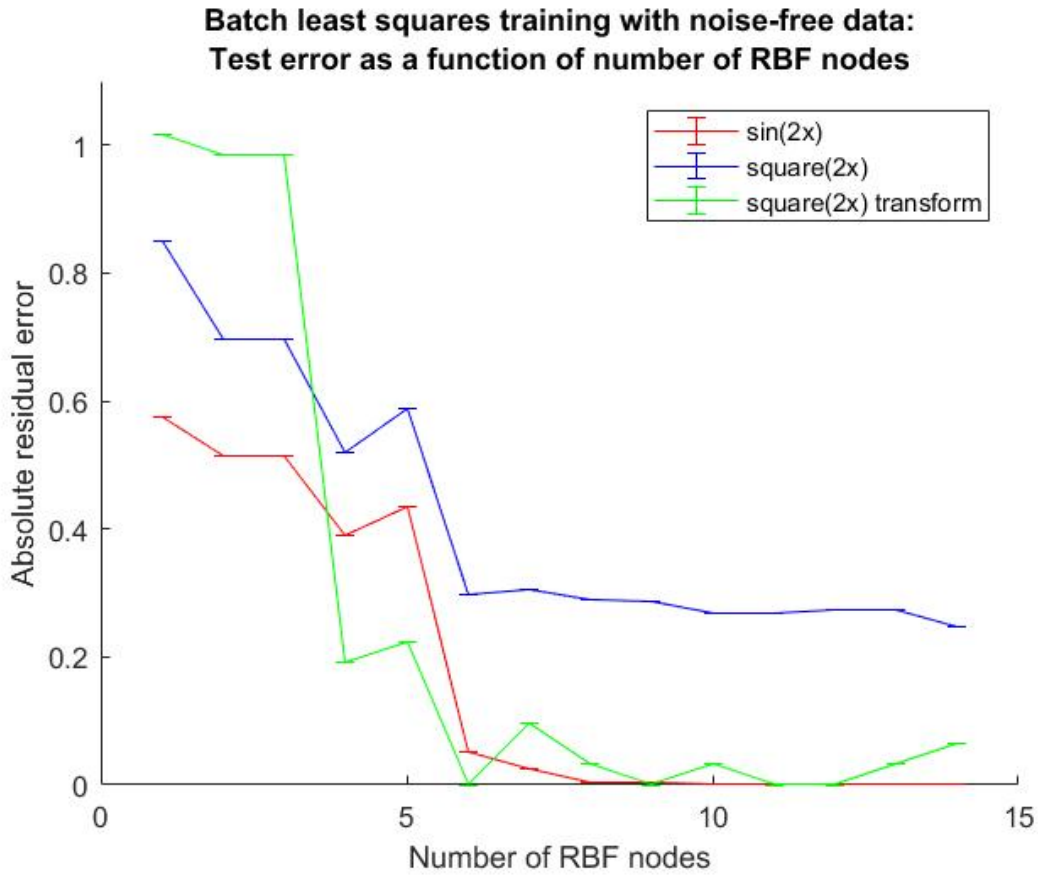

Figure 1: Test error as a function of n. Batch training and testing on clean data, σ=0.9.

Function approximation with RBF networks

- n = 6, 8 and 10 was needed to fulfill the error thresholds for the sin data.
- None of them was fulfilled for the square data.
- However, when rounding the square output to 1 or -1, only n=6 were needed for zero error. This transformation is convenient for classification.
- For decreasing σ and/or increasing n, the normal equations eventually result in *Matrix is close to singular or badly scaled*.

Regression with noise

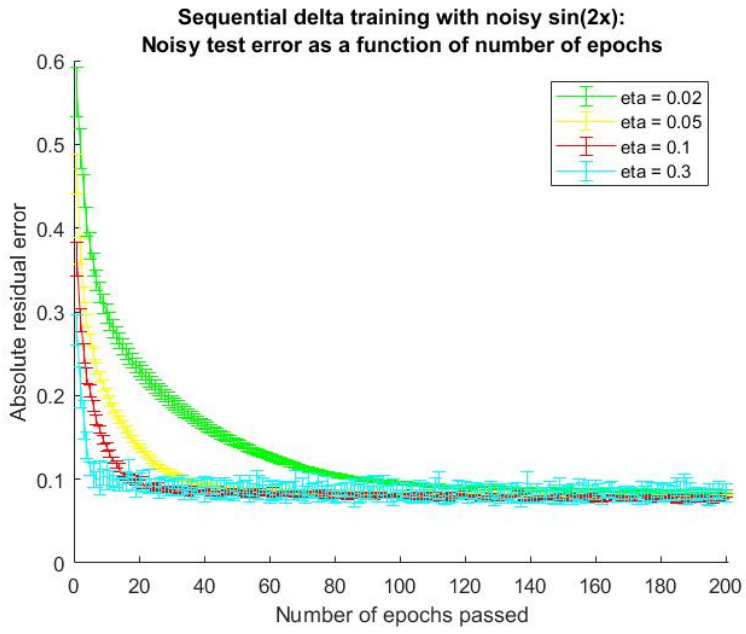

Figure 2: Test error as a function of number of epochs. Sequential training and testing on noisy sin(2x), n=10, σ=1.2.

- For sequential training, increasing η leads to a faster, but less stable algorithm (i.e. larger std).
- Batch learning does not depend on η, the optimal **w** is found in one step.

Regression with noise

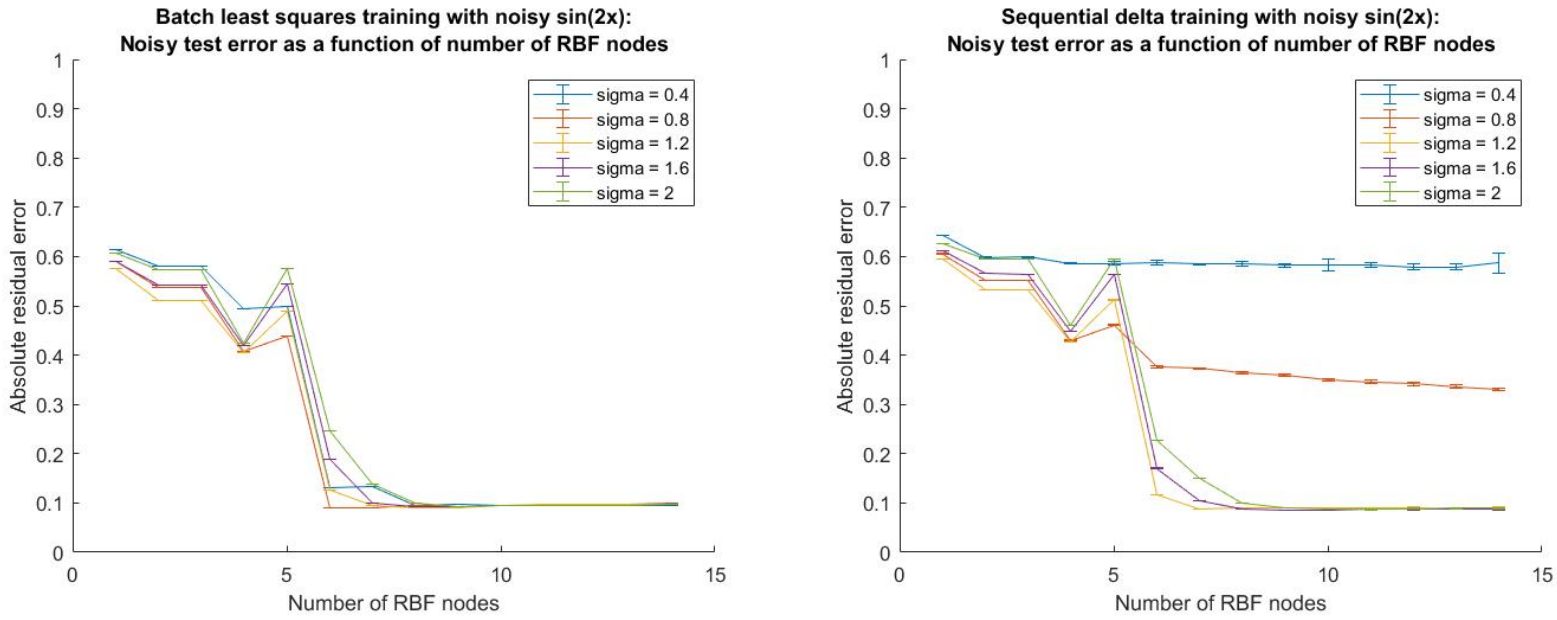

Figure 3: Test error as a function of n. Batch (left) vs sequential (middle) training and testing with noisy sin(2x).

- Figure 5 shows that best results for sin(2x) are obtained when $n \geq 8$.
- Sequential training improves significantly when $\sigma \geq 1.2$, while batch learning is almost unaffected.

Regression with noise

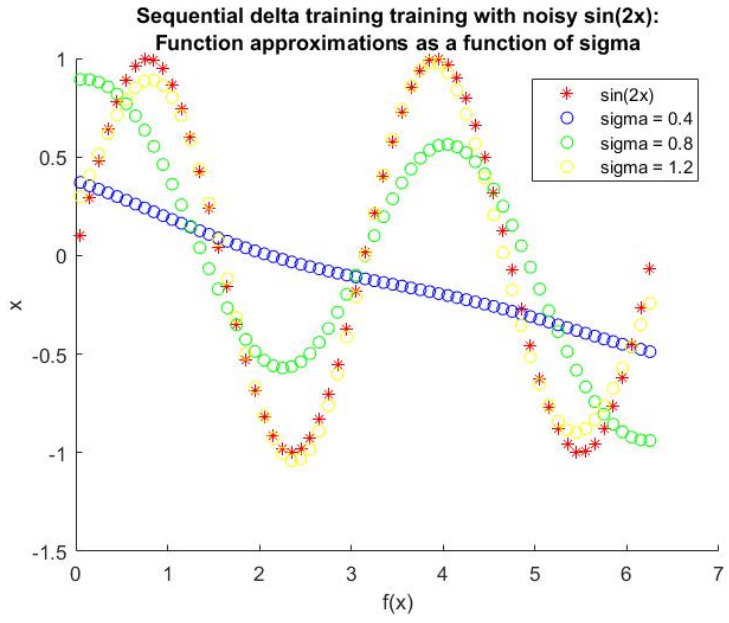

Figure 4: Function approximations of sin(2x). Sequential training with noisy data, n=9, η=0.02.

- For too small σ, the RBFs are too broad to make a good function approximation.

Regression with noise

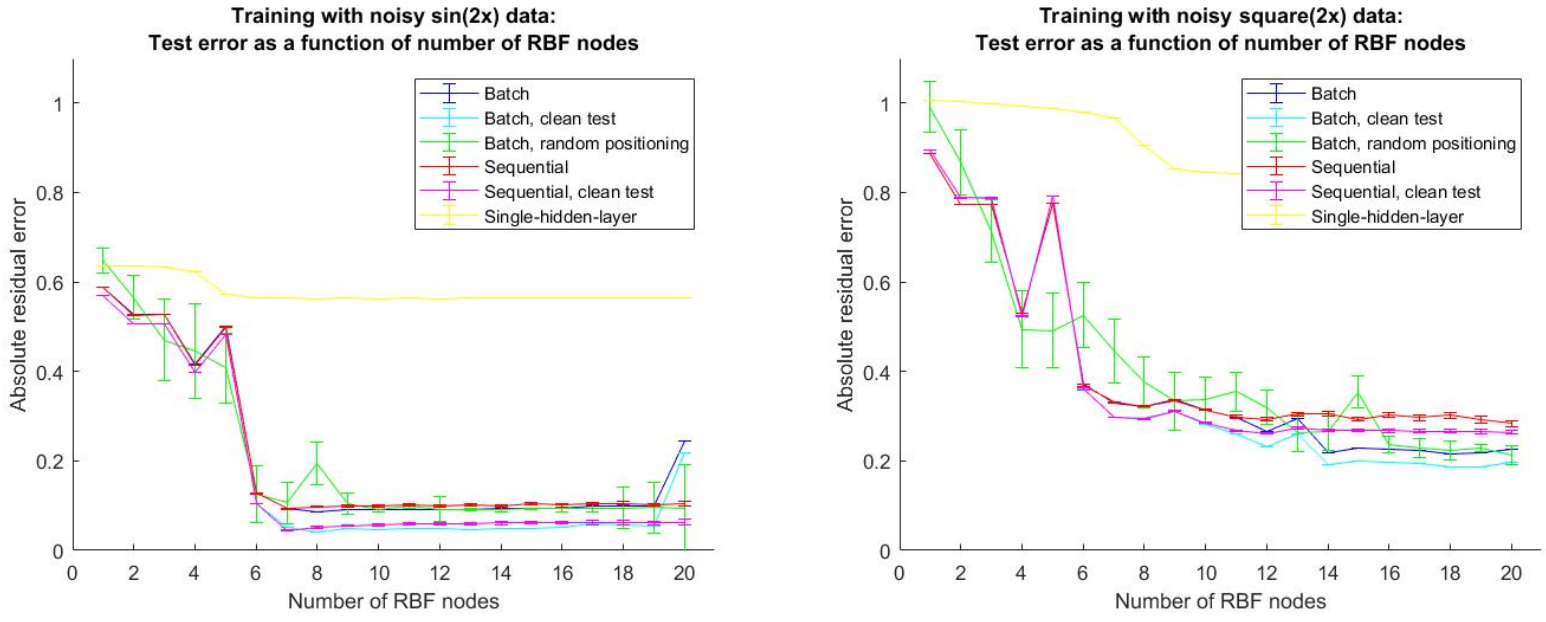

Figure 5: Test error as a function of n, η=0.05, α=0.5, σ=1.2 (left) or 2.0 (right).

Regression with noise

- We used uniform positioning, which perform well except for n=5 since all RBFs then are centered where f(x)=0. Random positioning gives less stability, but comparable results on average.
- Testing on clean data improves the results more for sin(2x) than square(2x), thus sin(2x) gives better generalisation performance.
- The single-hidden-layer perceptron performs worse than the RBF network, both in terms of generalisation performance and training time (batch converges in one step).

Competitive learning (CL) to initialize RBF units

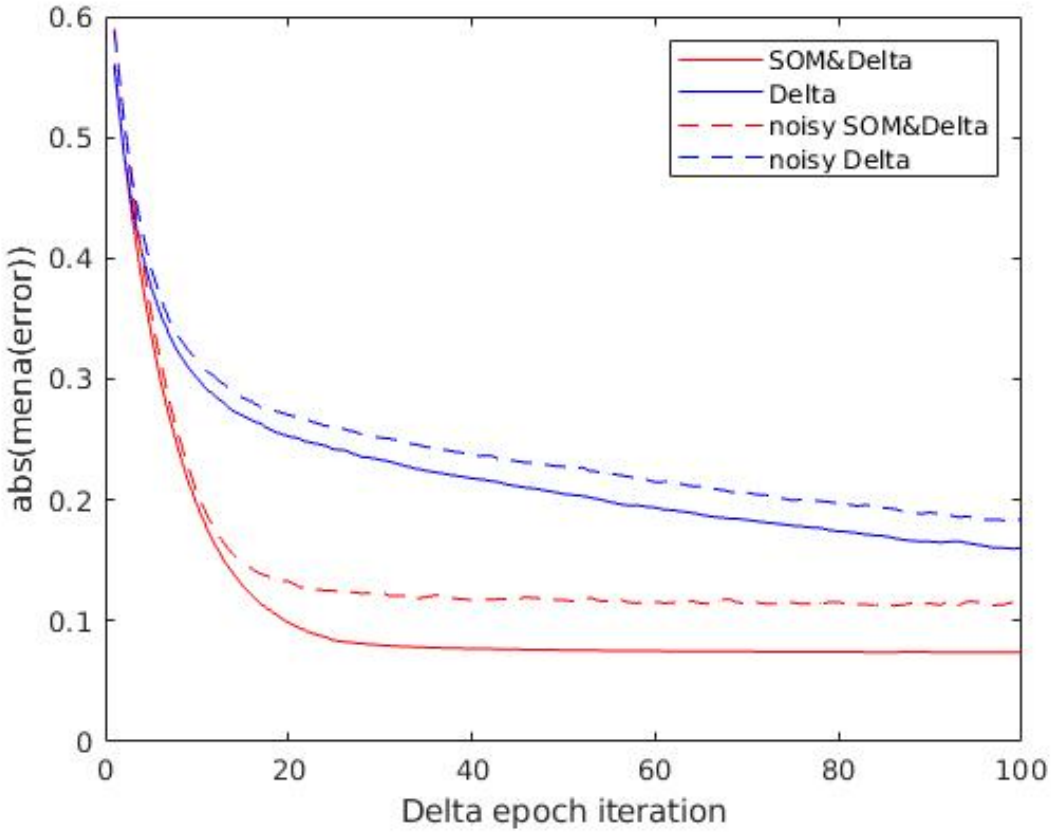

Figure 6: Delta method with and without CL for the positioning of RBF means, on clean and noisy datasets (ZERO neighbours).

Competitive learning (CL) to initialize RBF units

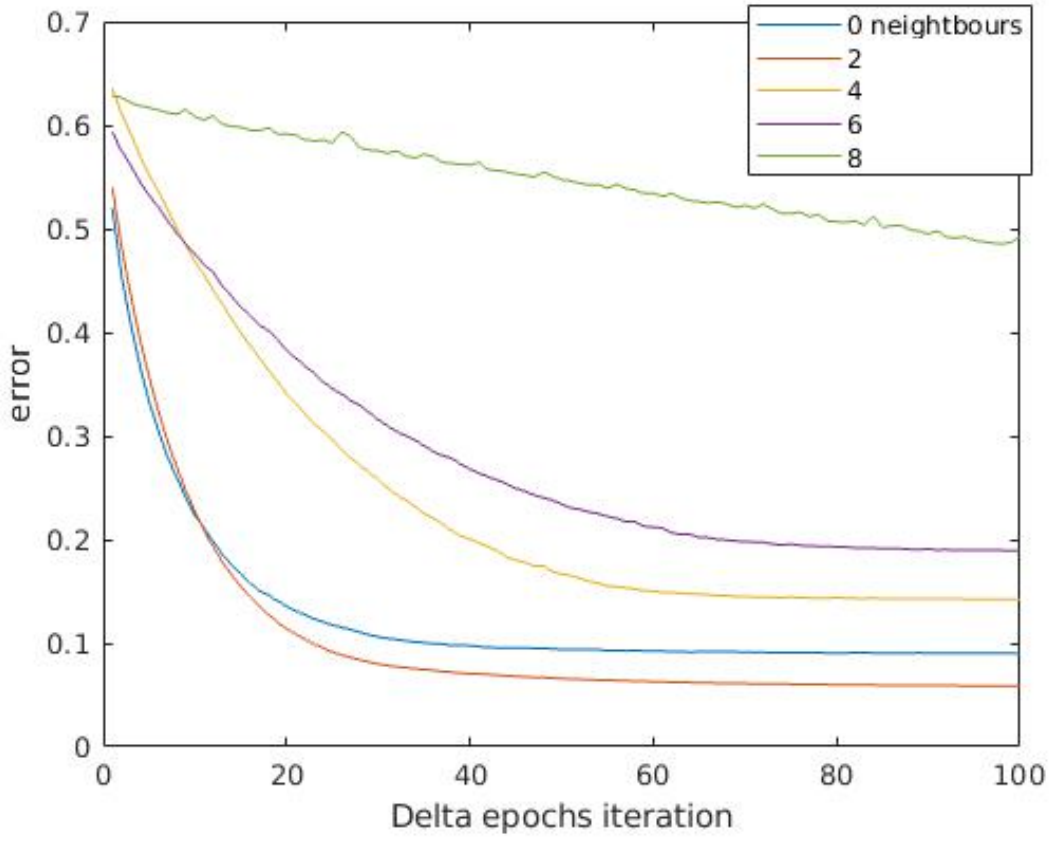

Figure 7: Updating two neighbours together with the winner can improve convergence, but using a large neighbourhood might slow it down: we risk to move RBF units in not-optimal positions.

CL + Delta. $f : \mathbb{R}^2 \to \mathbb{R}^2$

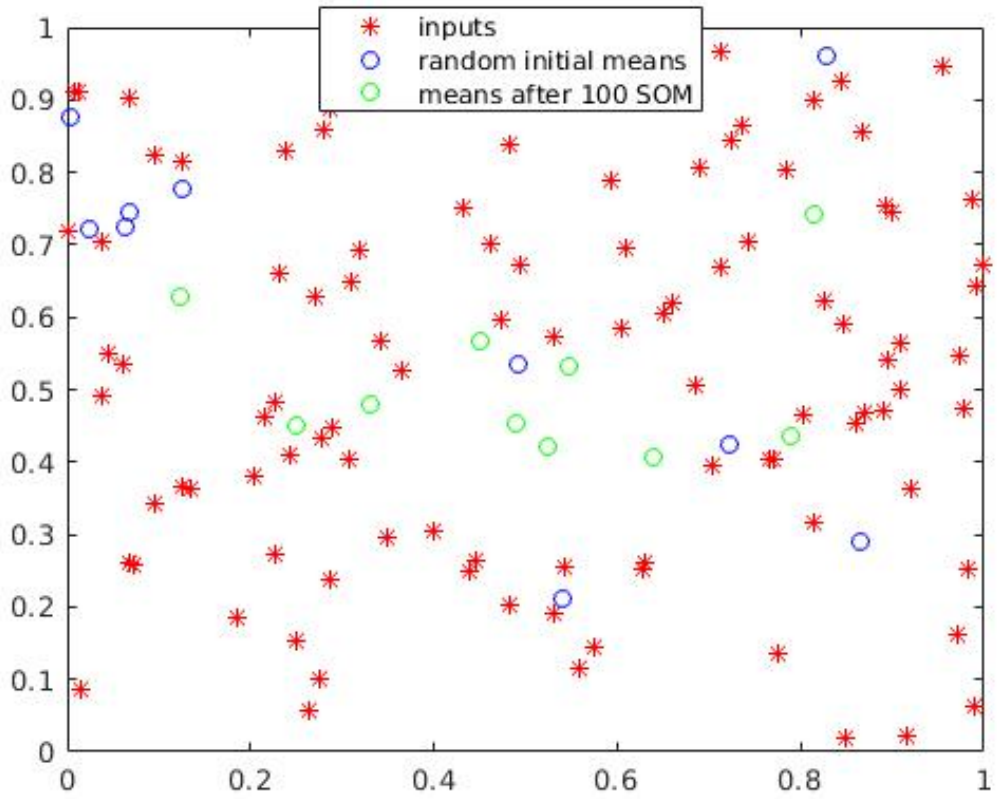

Figure 8: Train dataset, initial position of RBF means, final position after 100 CL iterations.

CL + Delta. $f : \mathbb{R}^2 \to \mathbb{R}^2$

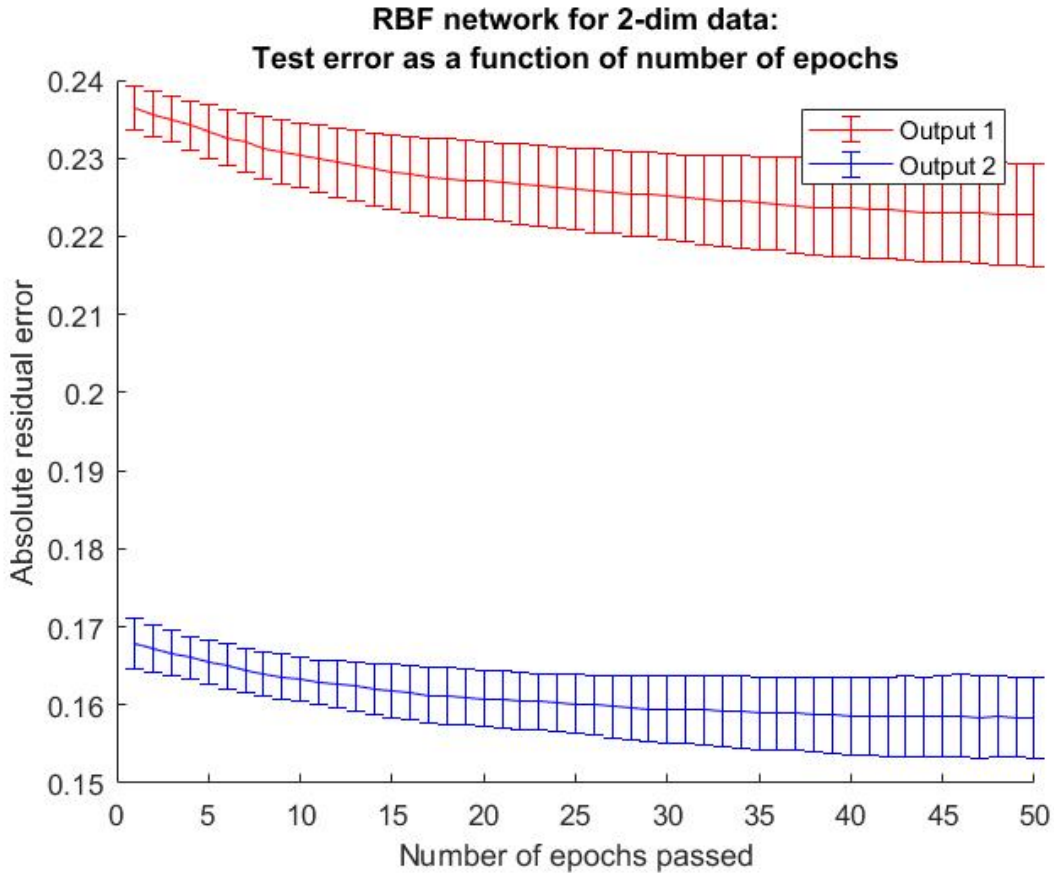

Figure 9: Test error decreases at each Delta epoch iteration for both outputs. Weights are updated independently for each dimension.

PART 2: (SOM) Topological Ordering of Animal Species

X=32x84
Sizes: $[25, 19, 16, 9, 4, 3, 2, 1, 0]$. With 20 epochs and $\eta = 0.2$ we got:

Neighbourhood: [25, 25, 19, 19, 16, 16, 9, 9, 4, 4, 3, 3, 2, 2, 1, 1, 1, 1, 0, 0, 0]

Result: [giraffe, camel, horse, pig, antelop, kangaroo, elephant, bat, lion, cat, rat, skunk, dog, hyena, bear, ape, rabbit, walrus, crocodile, seaturtle, frog, ostrich, penguin, pelican, duck, spider, grasshopper, beetle, butterfly, housefly, moskito, dragonfly]

Cyclic Tour

sizes = [2, 2, 1, 1, 1, 1, 1, 0, 0, 0]

We sorted the pos vector in the following way:

```
result_indices(pos)
      sort_pos = pos vector in increasing order
      result_indicies = []
      for every element in sort_pos, find its index in
      pos and add it to result_indices
```

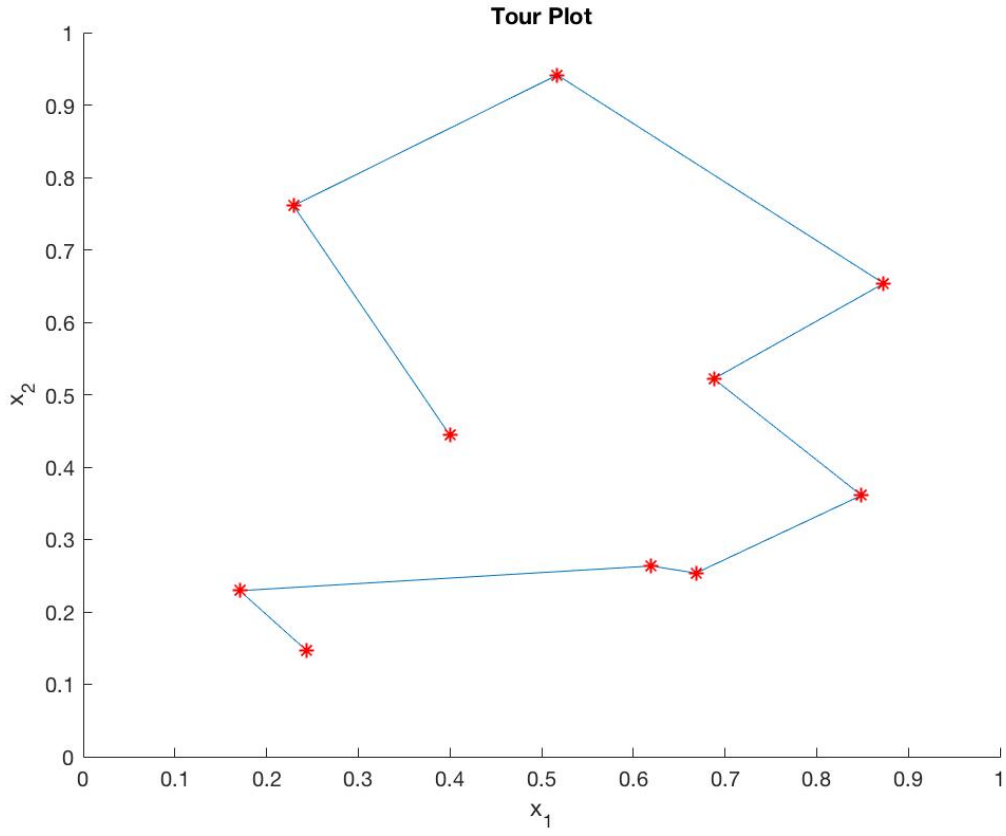

Figure 10: The SOM-algorithm finds a fairly short route which passes through all cities. But it is not guaranteed to always find the shortest route.

Data Clustering: Votes of MPs

X=349x31, sizes = [5, 4, 3, 2, 2, 1, 1, 1, 0, 0];

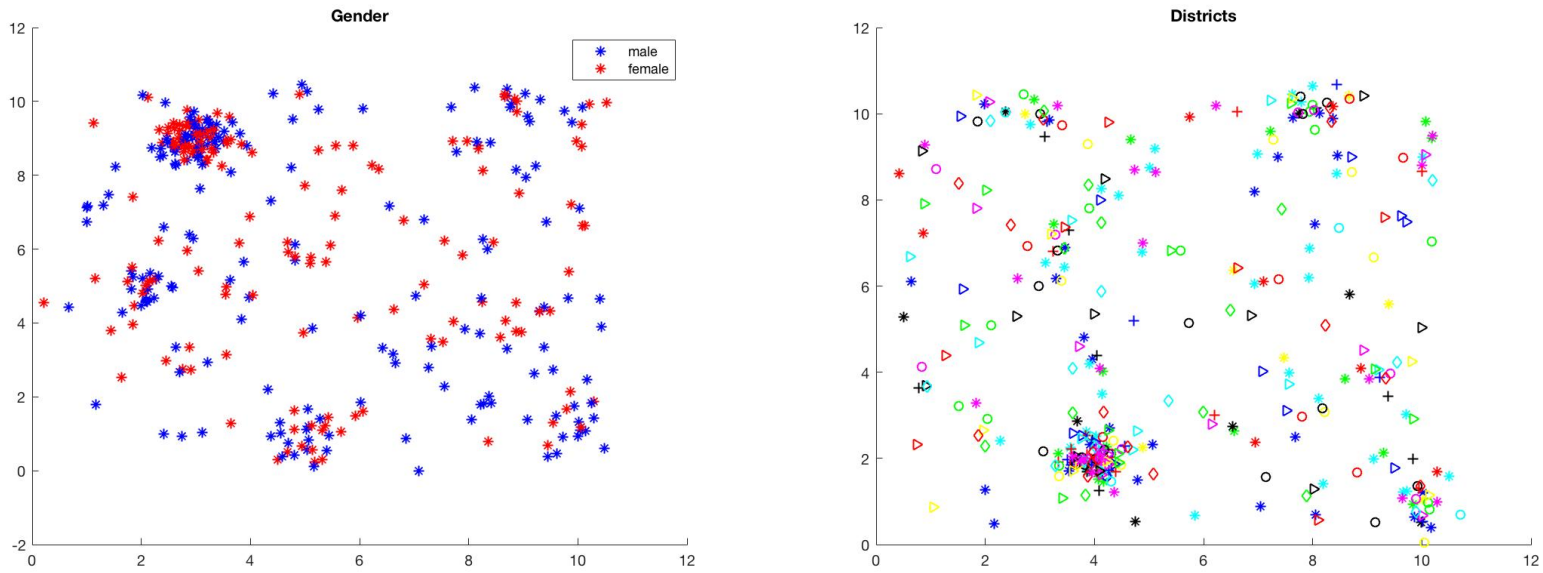

Figure 11: Results of the SOM-algorithm, 20 epochs and $\eta = 0.2$, on the `votes.dat` dataset with respect to Genders and Districts (29 different).

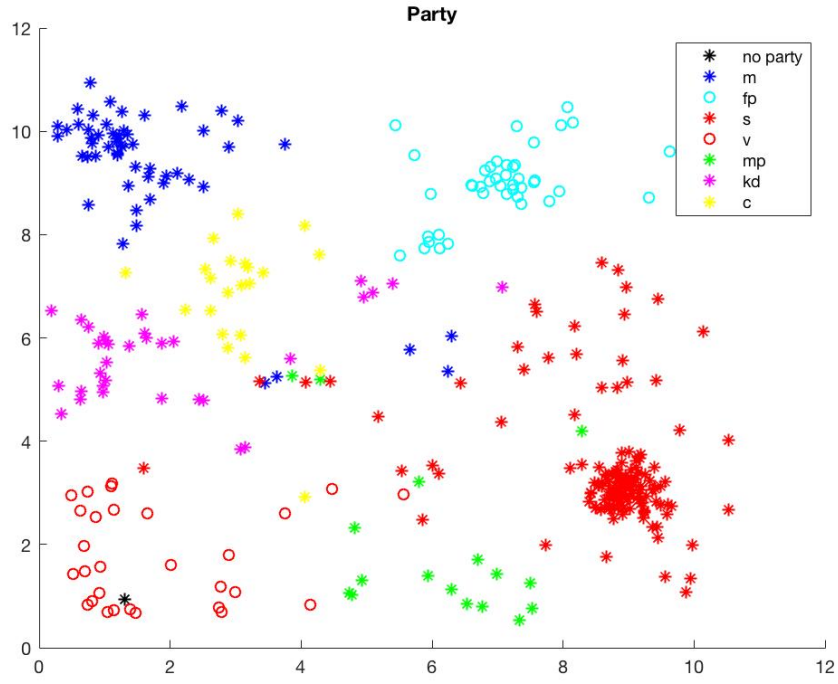

Figure 12: Results of the SOM-algorithm, 20 epochs and $\eta = 0.2$, on the `votes.dat` dataset with respect Parties (7 different + 1 "no party").