Exercise 18: Approximation with Neural Networks

Summer Term 2024

Approximation is another typical application area in which neural networks can yield good results. In this area, a function $f : R^n \rightarrow R^m$ maps an n-dimensional input domain onto an n-dimensional output domain. First, the function is defined by a set of points. Then, a neural network (or any other tool) should learn to approximate the function values in between reasonably well.

Review: Discuss the following questions:

1. What is an appropriate stopping criterion for this task?
2. How many connections w_{ij} does the network need?

To Do: In this exercise, you should implement a neural network that has to learn the simple two-dimensional function $f(x, y) = \cos(x) + \cos(y)$, i.e., $f : R^2 \rightarrow R$, in the range $(x, y) \in [-\pi..\pi]$.

Questions:

1. How many input and output units do you need?
2. What is the output range of a neuron, if you use the regular logistic transfer function?
3. How/where can you modify the number of network parameters?

Tasks: Reuse your simple backpropagation network that you have implemented in the previous exercise. Strip of the encoder stuff and loosely follow the following steps:

1. Introduce two parameters `l_pats` and `t_pats` that specify the number of learning and test patterns *per* input dimension. How many patterns do you need in total?
2. Introduce another parameter that specifies whether the training and test pattern should be generated randomly or systematically.
3. Complete the program and print both the learning and the test error.

   Hint: The documentation shows how you can directly call `gnuplot` from your own program.
4. Try to learn and generalize the given function reasonably well. In so doing, vary the learning rate η, the momentum α, the number of network parameters, the number of training patterns, the initialization mode, and the number of learning cycles. What can you observe?

Have fun, Theo and Ralf.