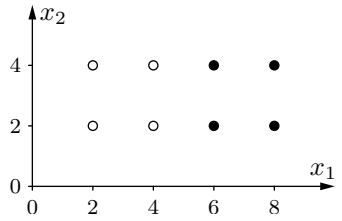
Artificial Neural Networks and Deep Learning Summer 2017
Christian Borgelt from 2017.06.27

Exercise Sheet 5

Exercise 28 Competitive Learning / Learning Vector Quantization

Consider the eight training data points that are shown on the right, which belong to two classes A (empty circles) and B (filled circles). This pattern set is to be quantized with the help of two reference vectors. Which final positon will be reached by these two reference vectors in an ideal case, if



a) only the "attraction rule" (patterns of the same class attract reference vectors),
b) both the "attraction rule" and the "repulsion rule" (patterns of different class repell reference vectors)

are used to change the positions of the reference vectors? (Hint: You need not calculate the steps of the procedure in detail. The solution can be read directly from the structure of the training patterns.)

Exercise 29 `xlvq/wlvq`: Competitive Learning / Learning Vector Quantization

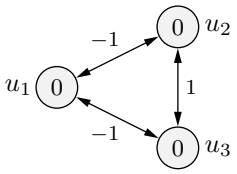
The programs that are available at `http://www.borgelt.net/lvqd.html` visualize learning vector quantization for two-dimensional data. (Higher-dimensional data may be loaded. In this case the program offers the option to select the two of the input variables that are to be used — see menu entry `Settings > Attributes`.) Apply the program to the Iris data with the input quantities petal length and petal width and let the program determine three reference vectors! (An input file in the appropriate format is contained in the source package.) Compare using the class information (one reference vector per class) with not using it (three reference vectors)!

Exercise 30 `xsom/wsom`: Self-organizing Maps

The programs that are available at `http://www.borgelt.net/somd.html` visualize the development of a self-organizing map, which is trained with points that are chosen randomly from a two-dimensional shape. Execute several training runs with this program! Vary the parameters, especially the shape of the area, from which the data points are chosen, the learning rate and the size of the neighborhood. What happens if the learning rate or the neighborhood radius is reduced too quickly ? (The learning rate is computed according to the formula $\eta(t) = \eta_0 t^{-\kappa}$. The values for η_0 (initial learning rate) and κ (learning rate decay exponent) can be entered in the parameter dialog box. The neighborhood radius is computed with an analogous formula.)

Exercise 31 Hopfield Networks: State Graph

The diagram on the right shows a simple Hopfield network. For this network, determine the final state(s) that are reached starting from the initial state $(\text{act}_{u_1}, \text{act}_{u_2}, \text{act}_{u_3}) = (-1, -1, -1)$!
(Hint: Use a state transition graph. In this graph, mark the initial state and the final state(s). Notice that there is no explicit start neuron and that therefore you have to consider all possible subsequent states.)



Exercise 32 Hopfield-Netze: Energy Function

Determine the energy function of the Hopfield network of Exercise 31! With the help of this energy function, compute the energies of the individual states of the network. Then arrange the states according to their energy, that is, draw a new state transition graph, in which the location of the states indicates their energy!

Exercise 33 Hopfield Networks: Pattern Recognition

We want to store the two patterns $(-1, +1, -1, +1)$ and $(+1, -1, -1, +1)$ in a Hopfield network with four neurons, that is, these two patterns are supposed to be the stable states of the network, which cannot be left, regardless of which neuron is updated.

a) Compute the connection weights and the threshold values of the neurons of a Hopfield network that stores the abovementioned patterns!

b) How many other patterns can be stored in this network?

c) Find two more patterns that could be stored in the network constructed in part a) without "forgetting" the old patterns! (Of course, in order to actually store these patterns, the connection weights may have to be modified.)

Exercise 34 `xhfn/whfn`: Pattern Recognition

The programs that are available at `http://www.borgelt.net/hfnd.html` visualize pattern recognition by a Hopfield network. Use this program to store a few patterns in the represented Hopfield network (for example, the seven digits and one block symbol used in the lecture, which are also avaliable in the file `hfnd/ex/numbers.hfn` in the source package). Afterward initialize the network randomly and observe which of the stored patterns is recognized. Are only the stored patterns recognized? Which other patterns are recognized as well/instead? How many patterns can be stored before reliable recognition breaks down?