Exercise Sheet 6

Exercise 34 Hopfield Networks: Solving Optimization Problems

We are given a sequence $F = (a_1, a_2, \ldots, a_n)$ of integer numbers. For simplicity, we assume that at least one of these numbers is non-negative. We desire to know the maximum partial sum of this sequence, that is, the maximum of sums of partial sequences of the sequence F, where by a partial sequence of the sequence F we mean a sequence $F_{ij} = (a_i, a_{i+1}, \ldots, a_j)$ with $1 \leq i \leq j \leq n$. That is, we desire to know

$$\mathrm{mts}(F) = \max_{1 \leq i \leq j \leq n} \sum_{k=i}^{j} a_k.$$

Construct a Hopfield network to solve this optimization problem!
(Hint: You have to find a suitable energy function.)

Exercise 35 Threshold Logic Units: Representation of Boolean Functions

In the lecture we studied an algorithm with which, given an arbitrary Boolean function $y = f(x_1, \ldots, x_n)$, a network of threshold logic units could be constructed that computed this function. This algorithm was based on representing the Boolean function in disjunctive normal form. Describe a dual algorithm that relies on representing the Boolean function in *conjunctive* normal form!

Exercise 36 Deep Learning: n-bit Parity

In the lecture it was shown how the n-bit parity function can be computed by a chain consisting of one *biimplication* network and $n-2$ *exclusive or* networks. Show how the n-bit parity function may also be computed by a binary tree of sub-networks, each of which computes the biimplication! How many layers does the resulting network have (as a function of n)? How many neurons does is contain in total (as a function of n)?

Exercise 37 Deep Learning: Autoencoder

As discussed in the lecture, an autoencoder is a 3-layer perceptron with as many output neurons as input neurons, which is supposed to map its inputs to (reconstructions of) its inputs. We assume that in the hidden layer as well as in the output layer the activation function is a rectified maximum (or ramp funtion) and that neither dropout training nor a restriction of the number of active neurons is used. In this case: why is it inappropriate to use as many neurons in the hidden layer as there are neurons in the input (or output) layer?
(Hint: Consider with which (very simple) parameters such a network may map its inputs entirely unchanged to its outputs.)

Exercise 38 Convolutional Neural Networks

In image processing grayscal images are often seen as a binary function $f(x, y)$ (x and y: coordinates of a pixel, function value: gray value of the pixel) and represented as a matrix. Consider the following image $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 10 & 10 & 0 \\ 0 & 10 & 10 & 10 & 10 & 0 \\ 0 & 10 & 10 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Convolutional Neural Networks are able to efficiently process image data independent of the position and rotation of objects that are visible in the images. For this so-called kernel functions are used, which are convolved with the image matrix. The computed *convolved features* are then used to recognize objects.

A possible convolved feature is the existence of an edge. If one assumes that at an edge the brightness changes significantly, edges may be detected as maxima of the first derivative of the image function f. This is the idea underlying the so-called Sobel operator, which consists of two sub-operators/kernel functions, namely

$$\mathbf{S}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \text{and} \quad \mathbf{S}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

a) Compute the convolved features $\mathbf{G}_x$ and $\mathbf{G}_y$ by sectionwise multiplication (convolution) of the operators $\mathbf{S}_x$ and $\mathbf{S}_y$ with the matrix $\mathbf{A}$, d.h. $\mathbf{G}_x = \mathbf{S}_x * \mathbf{A}$ and $\mathbf{G}_y = \mathbf{S}_y * \mathbf{A}$!

b) From the resulting direction-dependent matrices $\mathbf{G}_x$ and $\mathbf{G}_y$ a direction-independent matrix $\mathbf{G}$ is to be computed. For this the entries of $\mathbf{G}_x$ and $\mathbf{G}_y$ are squared, then summed, and finally the square root of the result is computed, that is, $\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$. Compute the matrix $\mathbf{G}$ with the help of the results obtained in part a) and describe the final result!

Exercise 40 Recurrent Neural Networks: Lissajous Curves

Lissajous or *Bowditch* curves are the graphs of a system of two parametric equations

$$x = a\sin(\omega_1 t + \delta) \qquad \text{and} \qquad y = b\sin(\omega_2 t).$$

Construct a recurrent neural network that computes (approximations of) Lissajous curves! How are the amplitudes a and b and the phase δ entered into the system? (Hint: Recall the recurrent neural network to compute the motion of a mass on a spring that was treated in the lecture.)