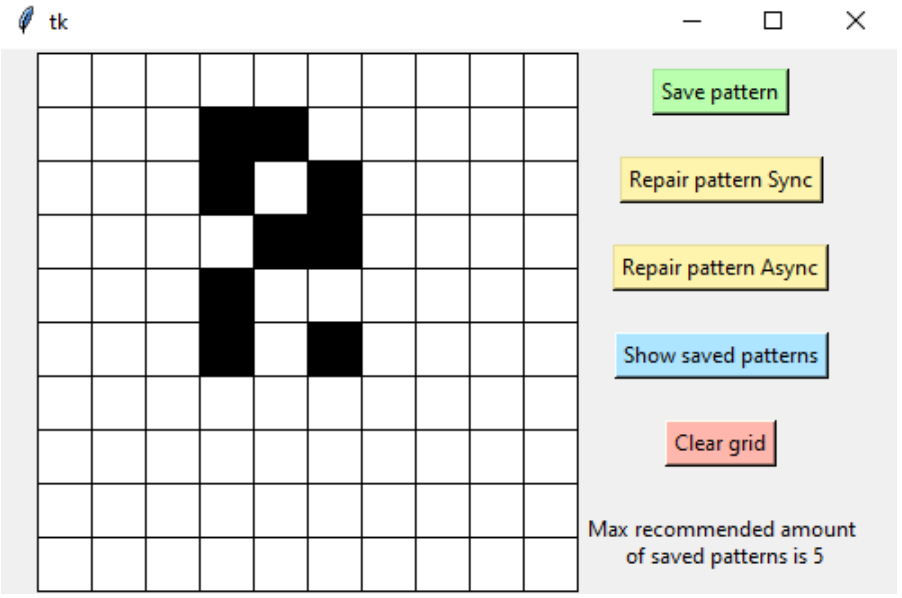
Task 3 - Hopfield network

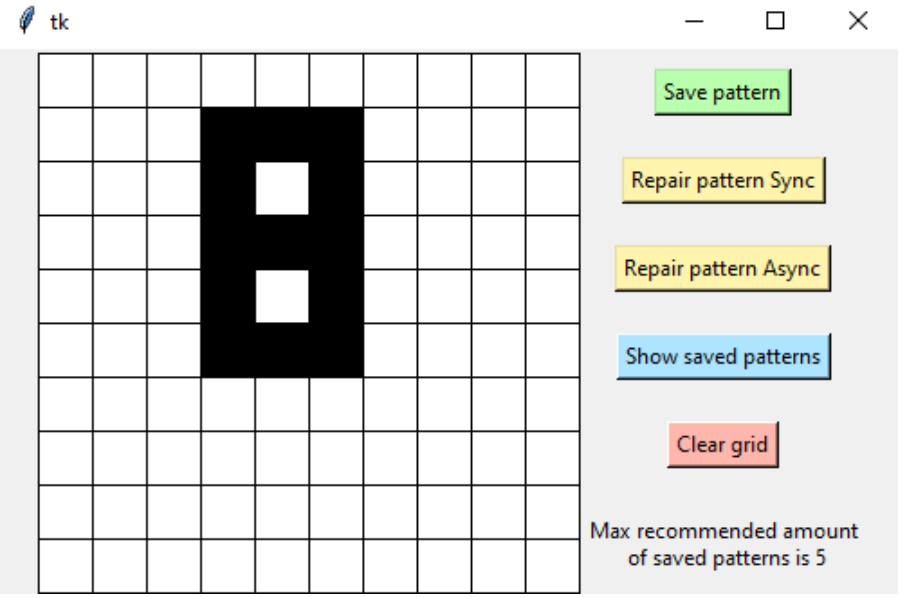
Use the Hopfield network to save and recover patterns.

Instructions

1. Design several (at least three) patterns. Patterns are given by a 2D figure.
2. Use a Hopfield network to save these patterns.
3. Change your patterns and use the Hopfield network to recover it - use synchronous as well as asynchronous recovery.



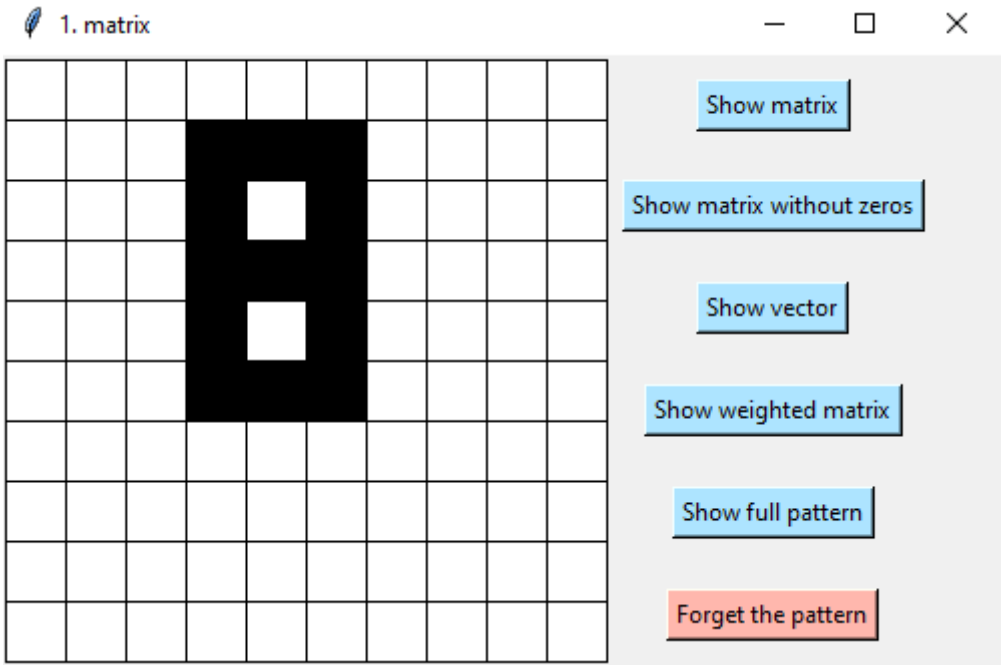
Before recovery



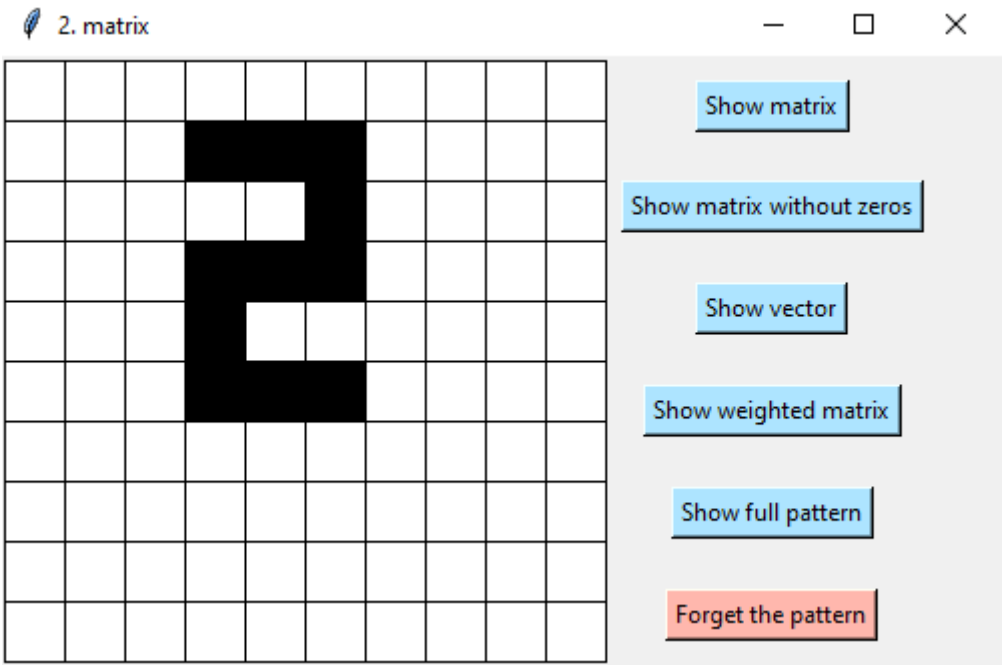
After recovery

Examples of patterns

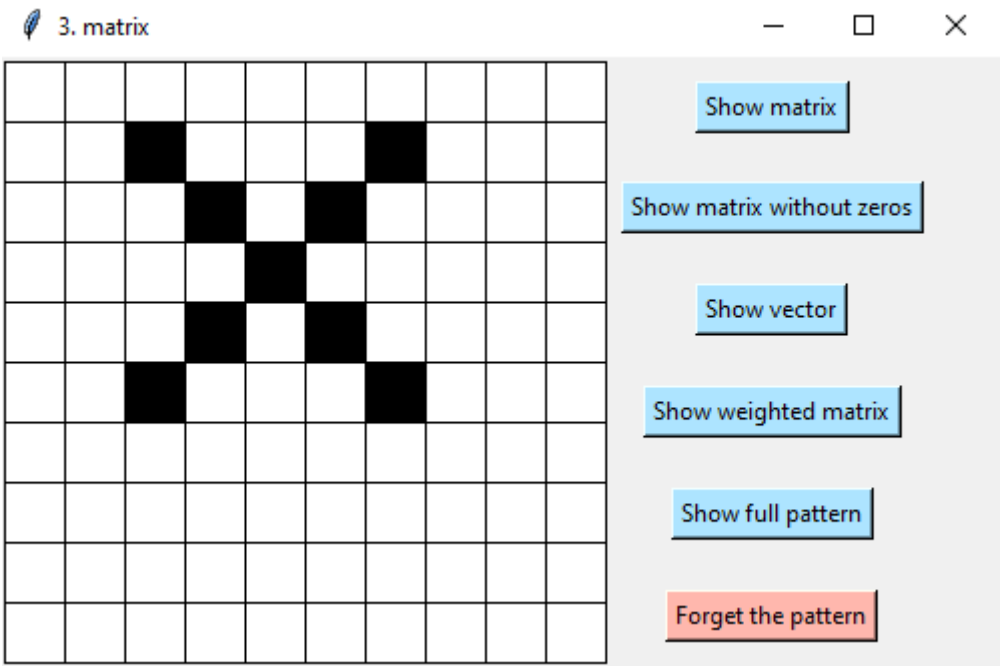
Original examples can be found here.



Example 1



Example 2



Example 3

Calculation example

Let's imagine an example from my diploma thesis (2.5 Hopfieldova síť) where we have a matrix *M*:

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}$$

Learning phase

In learning phase we have to calculate weight matrix *W* by dot product of *M* matrix with its transpose M^T:

$$W = M^T \cdot M = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Next step is to set zero values on diagonal by subtraction of unit matrix *I*:

$$W = W - I = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix}$$

Now we have our pattern saved in the memory of the hopfield neural network. If we would like to store multiple patterns we can do so by sum of matrices:

$$P = W_1 + W_2 = \begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ -1 & 1 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -2 & -0 & 0 \\ -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \end{bmatrix}$$

Asynchronous pattern recovery

Imagine, we have still the same weight matrix *W* and we would like to recover pattern from a following matrix *V*:

$$V = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix}$$

Rule for pattern recovery is, that if we apply the dot product on vectors' elements V_i and a column of matrix M_i we would get a following equation where if the result value is more than or equal to a specific threshold θ, the final result would be *1*, otherwise *-1*:

$$V_i = \begin{cases} +1 & \text{if } \sum_j W_{ij} V_j \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

For our example we will assume that the threshold $\theta = 0$ (it is a standard use case for Hopfield network). The recovery is done by activation of each neuron independently in a way, that we will do a dot product on the input vector *V* and the first column of the weight matrix *W* which will be our activation of the first neuron:

$$V_1 = sgn(V \cdot W_1) = sgn\left(\begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -1 \\ -1 \\ 1 \end{bmatrix} \right) = 1$$

The final result is used to change the value of the input vector on the position which we have selected from the weight matrix *W*, so in our example it is a first position. We will continue for the second column of the weight matrix *W*:

$$V_2 = sgn(V \cdot W_2) = sgn\left(\begin{bmatrix} 1 & 1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \\ -1 \end{bmatrix} \right) = -1$$

And if we keep continuing we will get our final pattern:

$$V_3 = sgn(V \cdot W_3) = sgn\left(\begin{bmatrix}1 & -1 & -1 & 1\end{bmatrix} \cdot \begin{bmatrix}-1\\1\\0\\-1\end{bmatrix}\right) = -1$$

$$V_4 = sgn(V \cdot W_4) = sgn\left(\begin{bmatrix}1 & -1 & -1 & 1\end{bmatrix} \cdot \begin{bmatrix}1\\-1\\-1\\0\end{bmatrix}\right) = 1$$

$$V = \begin{bmatrix}1 & -1 & -1 & 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

So as you can see the pattern has been restored and the pattern *V* is the same as pattern *M*.

Energy function

In my diploma thesis I have mentioned the energy function that is used from physics to simulate natural stop of pattern recovery. For our example we do not need to implement it, you can just use some type of checking, if the recovered pattern is still the same for some period of time (or iterations) and if so, you can stop. But if you really want to implement it, here is an equation:

$$E = -\frac{1}{2}\sum_{i,j=1}^{N} P_{ij}A_iA_j + \sum_{i=1}^{N}\theta_iA_i$$

More information can be found in the thesis, but the thesis is written only in Czech language.