20. Kohonen's algorithm

The update rule for the Kohonen network is:

$$\delta w_{ij} = \eta \Lambda(i,i_0)(x_j - w_{ij}) \ . \tag{1}$$

Here i_0 denotes the winning unit for pattern $\boldsymbol{x} = \Big(x_1, \ldots, x_N\Big)^{\mathsf{T}}$, η is the learning rate, and $\Lambda(i,i_0)$ is a Gaussian neighbourhood function

$$\Lambda(i,i_0) = \exp(-\frac{|\boldsymbol{r}_i - \boldsymbol{r}_{i_0}|^2}{2\sigma^2}) \tag{2}$$

with width σ. The vector $\boldsymbol{r}_i$ denotes the position of the i-th output neuron in the output array.

a. Explain the meaning of the parameter σ in Kohonen's algorithm. Discuss the nature of the update rule in the limit of $\sigma \to 0$.

Answer. The parameter σ regulates the width of the neighbouring function $\Lambda(i,i_0)$. This function is centred at the winning neuron i_0 ($\Lambda(i_0,i_0) = 1$), and it decreases as the distance from this neuron increases. The update rule (1) assures that the weight assigned to the winning neuron i_0 is moved by a fraction η towards the fed pattern $\boldsymbol{x}$. Moreover, the neighbourhood function assures that the weights assigned to the remaining neurons are also moved towards $\boldsymbol{x}$ with a fraction determined by the width of $\Lambda(i,i_0)$: the fraction is larger the closer the neuron i is to the winning neuron i_0. In the limit of $\sigma \to 0$, the update rule (1) reduces to the simple-competitive learning, because only the winning neuron for pattern $\boldsymbol{x}$ is updated in this case. In other words, the rule becomes:

$$\delta w_{ij} = \begin{cases} \eta(x_j - w_{ij}), & \text{for } i = i_0 \,, \\ 0, & \text{otherwise} \,. \end{cases} \tag{3}$$

b. Discuss and explain the implementation of the Kohonen's algorithm.

Answer. M output neurons arranged in a d-dimensional lattice. The position of the i^{th} output neuron in the output space is a d-dimensional vector $\boldsymbol{r}_i$ (each $\boldsymbol{r}_i$ is fixed). Weight matrix $\mathbb{W}$ is an MxN matrix with elements w_{ij} ($i = 1, \ldots, M$, $j = 1, \ldots, N$). The weight vector $\boldsymbol{w}_i$ assigned to the output neuron i is N-dimensional (as are the input patterns $\boldsymbol{x}$): $\boldsymbol{w}_i = (w_{i1}, \ldots, w_{i,N})^{\mathsf{T}}$. Learning consists of two phases: ordering and convergence phase. Usually, ordering phase lasts for $T_{\text{order}} \approx 1000$ updates. The convergence phase starts after the ordering phase is finished. Initialisation: weights are initialised to random numbers from a relevant interval for a given problem, e.g. from $[-1,1]$. Set the current update number t to $t = 0$. Learning:

1. Update $t \leftarrow t + 1$.
2. Draw a random pattern $\boldsymbol{x}$ from the set of input patterns.
3. Competition: find a neuron i_0 such that $\boldsymbol{w}_{i_0}$ is closest to $\boldsymbol{x}$ in the Euclidian sense, i.e.

$$|\boldsymbol{x} - \boldsymbol{w}_{i_0}| \leq |\boldsymbol{x} - \boldsymbol{w}_i| \text{ for all } i \ . \tag{4}$$

4. Update weight vectors (note that the learning rate η and the neighbourhood function $\Lambda(i,i_0)$ depend on the update number t as explained below)

$$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i + \boldsymbol{\delta w_i} \ , \tag{5}$$

where

$$\delta w_{ij} = \eta(t)\Lambda(i,i_0,t)(x_j - w_{ij}) \ , \text{ and } \Lambda(i,i_0,t) = \exp(-\frac{|\boldsymbol{r}_i - \boldsymbol{r}_{i_0}|^2}{2\sigma(t)^2}) \ . \tag{6}$$

The dependence on the update number is as follows: if $t < T_{\text{order}}$, then $\sigma(t) = \sigma_0 \exp(-\frac{t}{\tau})$, and $\eta(t) = \eta_0 \exp(-\frac{t}{\tau})$, where $\tau \approx \frac{T_{\text{order}}}{\ln(\sigma_0)}$, σ_0 is set to the largest distance in the output lattice, and η_0 is set to a small (but not too small) value, usually $\eta_0 = 0.1$. When η_0 is not too small, the network is likely to get rid of *kinks* during the ordering phase. If, however, $t > T_{\text{order}}$ (convergence phase; fine tunning), then both η and σ are kept constant and small (usually, $\eta = 0.01$, and $\sigma \approx 1$).
5. Repeat steps 1-4 until convergence.

21. Radial basis functions

XOR problem.

a. Show that this problem cannot be solved by a simple perceptron.

Answer. A problem is solvable by a simple perceptron if it is linearly separable. For the two-dimensional XOR problem (Fig. 1) this means one must find a plane separating the two types of target outputs. From Fig. 1 we see that no such plane can be found: there will always be at least one point that will be misclassified (the point on the “wrong side” of the plane). An example is shown by a dashed line in Fig. 1 - on the right side from the plane, there is only one pattern with the target output 1. On the opposite side of the plane there are two patterns with target output 0, but also one pattern with the target output 1. The latter pattern would be misclassified as the pattern with the output 0.

b. Solve the problem by transforming the input space using radial-basis functions.

Answer. Applying the radial-basis functions suggested in the task, we find that the input patterns have the following coordinates in the transformed space $(g_1,g_2)^{\mathsf{T}}$:

$$\begin{aligned} g_1(\boldsymbol{x}^{(1)}) &= \exp(-|(1,1)^{\mathsf{T}} - (1,1)^{\mathsf{T}}|^2) = \exp(0) = 1 \ , \\ g_2(\boldsymbol{x}^{(1)}) &= \exp(-|(1,1)^{\mathsf{T}} - (0,0)^{\mathsf{T}}|^2) = \exp(-2) \approx 0.14 \ . \end{aligned} \tag{7}$$

$$\begin{aligned} g_1(\boldsymbol{x}^{(2)}) &= \exp(-|(1,0)^{\mathsf{T}} - (1,1)^{\mathsf{T}}|^2) = \exp(-1) \approx 0.37 \ , \\ g_2(\boldsymbol{x}^{(2)}) &= \exp(-|(1,0)^{\mathsf{T}} - (0,0)^{\mathsf{T}}|^2) = \exp(-1) \approx 0.37 \ . \end{aligned} \tag{8}$$

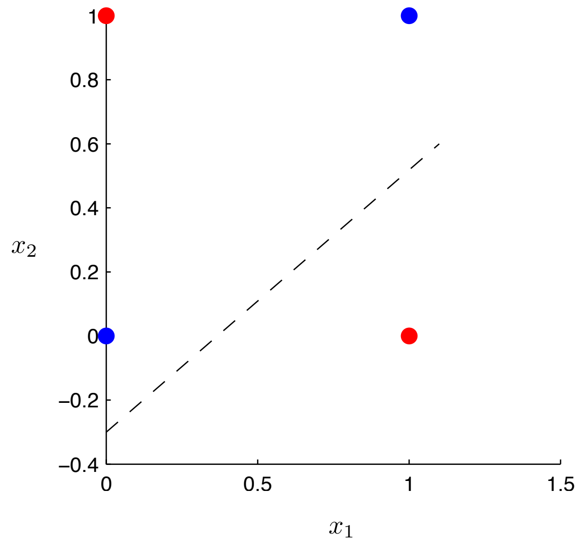

Figure 1: Question 21b: XOR problem. Input patterns (circles) in input space $(x_1,x_2)^\mathsf{T}$. Colours denote target output for a given input pattern: target output 0 is denoted by blue, and target output 1 is denoted by red. Dashed line is an example of a line that tries to solve the XOR problem. In this case, the input $(0,1)^\mathsf{T}$ would be misclassified as having the output 0.

$$\begin{aligned} g_1(\boldsymbol{x}^{(3)}) &= \exp(-|(0,1)^\mathsf{T} - (1,1)^\mathsf{T}|^2) = \exp(-1) \approx 0.37\ , \\ g_2(\boldsymbol{x}^{(3)}) &= \exp(-|(0,1)^\mathsf{T} - (0,0)^\mathsf{T}|^2) = \exp(-1) \approx 0.37\ . \end{aligned} \tag{9}$$

$$\begin{aligned} g_1(\boldsymbol{x}^{(4)}) &= \exp(-|(0,0)^\mathsf{T} - (1,1)^\mathsf{T}|^2) = \exp(-2) \approx 0.14\ , \\ g_2(\boldsymbol{x}^{(4)}) &= \exp(-|(0,0)^\mathsf{T} - (0,0)^\mathsf{T}|^2) = \exp(0) = 1\ . \end{aligned} \tag{10}$$

Note that both input patterns ($\boldsymbol{x}^{(2)}$ and $\boldsymbol{x}^{(3)}$) are transformed to the same vector in the transformed input space $(g_1,g_2)^\mathsf{T}$ (see the red point in Fig. 2). Applying a simple perceptron to the transformed input patterns (in space $(g_1,g_2)^\mathsf{T}$), the problem is linearly separable. Indeed, there is a *decision boundary* separating the two classes of data: in this case, the network output at the decision boundary is equal to 0.5 (because the activation function is linear)

$$\boldsymbol{w}^\mathsf{T}\boldsymbol{g} - \theta = 0.5, \text{ for } \boldsymbol{g} \text{ at the decision boundary}\ , \tag{11}$$

where $\boldsymbol{w} = (w_1,w_2)^\mathsf{T}$. [Note: if the target outputs are $+1$ or -1, then we would require the sigmoid activation function, and that the network output at the boundary is equal to zero.] The weights and the threshold for the simple perception corresponding to the decision boundary shown in Fig. 2 are $w_1 = -1$, $w_2 = -1$, $\theta = -1.5$. [Note: there are other possible solutions to Eq. (11). Particularly, the weight vector $-\boldsymbol{w}$ and the corresponding threshold is also a possible solution. One needs to check the network output for the problem given to decide which solution to take. In this task, one requires that the network outputs for the two blue points are smaller than 0.5, whereas for the red point, the network output is required to be larger than 0.5.]

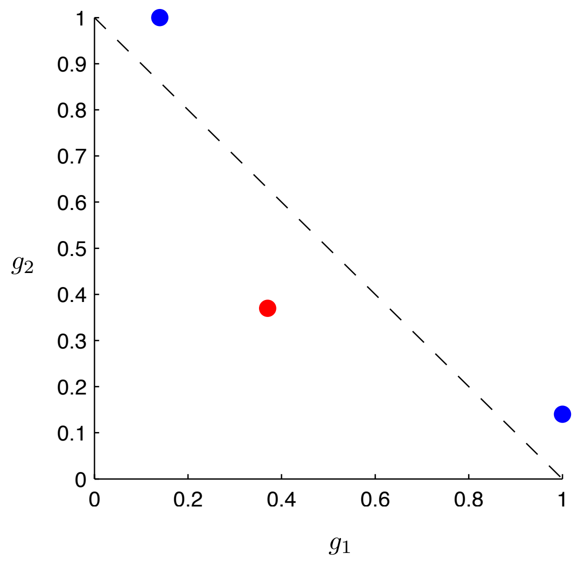

Figure 2: Question 21b: XOR problem in transformed space. Input patterns (circles) in space $(g_1,g_2)^{\mathsf{T}}$. Colours denote target output for a given input pattern: target output 0 is denoted by blue, and target output 1 is denoted by red. Dashed line is an example of a line solving the XOR problem (decision boundary). In this case, the weight vector and the threshold corresponding to the decision boundary shown are $\boldsymbol{w} = (-1, -1)^{\mathsf{T}}$ and $\theta = -1.5$.