CHALMERS, GÖTEBORGS UNIVERSITET

RE-EXAM for
ARTIFICIAL NEURAL NETWORKS

COURSE CODES: **FFR 135, FIM 720 GU, PhD**

Time:	January 4, 2022, at $14^{00} - 18^{00}$
Place:	Johanneberg
Teachers:	Bernhard Mehlig, 073-420 0988 (mobile)
	Ludvig Storm, visits at 14^{30} and 17^{30}
Allowed material:	Mathematics Handbook for Science and Engineering
Not allowed:	Any other written material, calculator

Maximum score on this exam: 12 points.
Maximum score for homework problems: 12 points.
To pass the course it is necessary to score at least 5 points on this written exam.
CTH >13.5 passed; >17 grade 4; >21.5 grade 5,
GU >13.5 grade G; > 19.5 grade VG.

---

1. Hopfield model with time-continuous dynamics. Consider a Hopfield net with continuous-time dynamics:

$$\tau \tfrac{\mathrm{d}}{\mathrm{d}t} n_i = -n_i + g\Big(\sum_j w_{ij} n_j - \theta_i\Big)$$

with $g(b) = (1+\mathrm{e}^{-b})^{-1}$ and time scale τ. Show that the energy function

$$E = -\tfrac{1}{2}\sum_{ij} w_{ij} n_i n_j + \sum_i \theta_i n_i + \sum_i \int_0^{n_i} \mathrm{d}n\, g^{-1}(n)$$

cannot increase under the network dynamics if the weights are symmetric. Here g^{-1} is the inverse function of g, so that $g^{-1}\big(g(b)\big) = b$. *Hint:* use the fact that $g(b)$ is a monotonically increasing function of b. (**2**p).

Solution: We want to show that $\frac{\mathrm{d}}{\mathrm{d}t}E \leq 0$. Differentiating E w.r.t time t yields

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -\sum_{ij} w_{ij} \frac{\mathrm{d}b_i}{\mathrm{d}t} g'(b_i) g(b_j) + \sum_i \theta_i \frac{\mathrm{d}b_i}{\mathrm{d}t} g'(b_i) + \sum_i \frac{\mathrm{d}b_i}{\mathrm{d}t} b_i g'(b_i) \qquad (1)$$

$$= \sum_i \frac{\mathrm{d}b_i}{\mathrm{d}t} g'(b_i)\Big[-\sum_j w_{ij} g(b_j) + \theta_i + b_i\Big], \qquad (2)$$

where we used $w_{ij} = w_{ji}$ in the first step. The equation of motion of b_i reads

$$\tau \frac{\mathrm{d}b_i}{\mathrm{d}t} = \tau \sum_j w_{ij} \frac{\mathrm{d}n_i}{\mathrm{d}t} = -b_i - \theta_i + \sum_j w_{ij} g(b_j)\,. \tag{3}$$

Combining Equations (1) and (3) yields

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -\frac{1}{\tau} \sum_i g'(b_i) \Big[b_i + \theta_i - \sum_j w_{ij} g(b_j) \Big]^2 . \tag{4}$$

This expression cannot be positive because $g'(b) > 0$, and this means that local minima of E are attractors. The steady states $\boldsymbol{n}^*$ of the dynamics satisfy $n_i^* = g(b_i^*)$, $b_i^* = \sum_j w_{ij} n_i^* - \theta_i$, and $\mathrm{d}E^*/\mathrm{d}t = 0$. But note that they need not be attractors. Consider an example: $w_{11} = w_{22} = 0$, $w_{12} = w_{21} = 10$, and $\theta_1 = \theta_2 = -5$. The steady state $\boldsymbol{n}^* = [\frac{1}{2}, \frac{1}{2}]^{\mathsf{T}}$ is a saddle point. See Exercise 9.2 and Equation (9.18a) in the course book.

2. Three-point probabilities in 3×3 bars-and-stripes data set. Demonstrate that a Boltzmann machine requires hidden units to learn the 3×3 data set shown in Figure 1(**a**). To this end, evaluate all three-point probabilities $P(x_1 = \pm 1, x_2 = \pm 1, x_3 = \pm 1)$ as shown in panel (**b**). Here $x_j = +1$ represents ■, and $x_j = -1$ stands for □. Check whether these three-point probabilities factorise. For example, does

$$P(x_1{=}1, x_2{=}1, x_3{=}-1) \neq P(x_1{=}1, x_2{=}1) P(x_3{=}-1)$$

hold or not? Use your results to explain why a Boltzmann machine needs hidden units to learn the data set (**a**). Now consider the data set in Figure 1(**c**), only stripes. Explain why no hidden units are needed for (**c**). (2**p**).

Solution: The eight three-point probabilities $P(x_1{=}\pm 1, x_2{=}\pm 1, x_3{=}\pm 1)$ for the data set are listed in Table 1. Since $P(x_1 = 1, x_2 = 1) = \frac{5}{14}$ and $P(x_3 = -1) = \frac{7}{14}$, we see that $P(x_1{=}1, x_2{=}1, x_3{=}-1) \neq P(x_1{=}1, x_2{=}1) P(x_3{=}-1)$. This three-point probability does not factorise. More generally, Table 1 shows that the three-point probability $P(x_1, x_2, x_3)$ does not factorise as $P(x_1, x_2) P(x_3)$ if x_1 and x_2 have the same colour, because in this case one cannot say from x_1 and x_2 alone whether a pattern should be classified as bars or stripes. As a consequence, a Boltzmann machine requires hidden units to represent the data set (**a**). For the data set (**c**), by contrast, the three-point probabilities do factorise. For example, $P(x_1{=}1, x_2{=}1, x_3{=}-1) = \frac{1}{8}$, $P(x_1{=}1, x_2{=}1) = \frac{1}{4}$, and $P(x_3{=}-1) = \frac{1}{2}$. Since the three-point correlations can be expressed in terms of two-point correlations, no hidden units are needed to represent this data set with a Boltzmann machine.

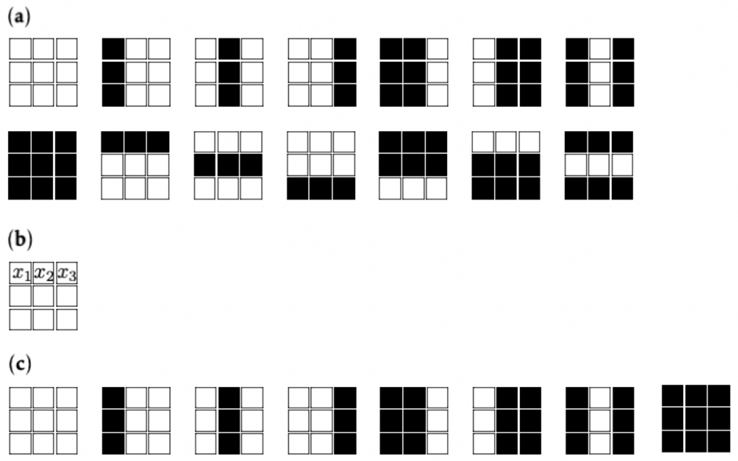

Figure 1: (**a**) 3×3 bars-and-stripes data set. The shown patterns occur with probability $P_{\text{data}} = \frac{1}{14}$, all other patterns have $P_{\text{data}} = 0$. (**b**) Definition of the bits x_1, x_2, and x_3. (**c**) Data set with stripes only. The shown patterns occur with probability $P_{\text{data}} = \frac{1}{8}$, all other patterns have $P_{\text{data}} = 0$.

Table 1: Three-point probabilities for the data set shown in Figure 1. Question 2.

x_1	x_2	x_3	$P(x_1, x_2, x_3)$	$P(x_1, x_2)$	$P(x_3)$
-1	-1	-1	$\frac{4}{14}$	$\frac{5}{14}$	$\frac{1}{2}$
1	-1	-1	$\frac{1}{14}$	$\frac{2}{14}$	$\frac{1}{2}$
-1	1	-1	$\frac{1}{14}$	$\frac{2}{14}$	$\frac{1}{2}$
-1	-1	1	$\frac{1}{14}$	$\frac{5}{14}$	$\frac{1}{2}$
1	1	-1	$\frac{1}{14}$	$\frac{5}{14}$	$\frac{1}{2}$
-1	1	1	$\frac{1}{14}$	$\frac{2}{14}$	$\frac{1}{2}$
1	-1	1	$\frac{1}{14}$	$\frac{2}{14}$	$\frac{1}{2}$
1	1	1	$\frac{4}{14}$	$\frac{5}{14}$	$\frac{1}{2}$

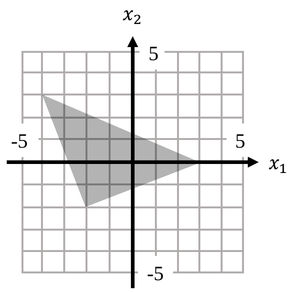

Figure 2: Linearly inseparable classification problem, Question 3.

3. Linearly inseparable problem. A classification problem is given in Figure 2. Inputs $\boldsymbol{x}^{(\mu)}$ inside the gray triangle have targets $t^{(\mu)} = 1$, inputs outside the triangle have targets is $t^{(\mu)} = 0$. The problem can be solved by a perceptron with one hidden layer with three neurons $V_j^{(\mu)} = \theta_{\rm H}\left(-\theta_j + \sum_{k=1}^{2} w_{jk} x_k^{(\mu)}\right)$, for $j = 1, 2, 3$. The network output is computed as $O^{(\mu)} = \theta_{\rm H}\left(-\Theta + \sum_{j=1}^{3} W_j V_j^{(\mu)}\right)$. Here $\theta_{\rm H}(b)$ is the Heaviside function:

$$\theta_{\rm H}(b) = \begin{cases} 1 \text{ if } b > 0 \\ 0 \text{ otherwise} \end{cases}$$

Find weights w_{jk}, W_j and thresholds θ_j, Θ that solve the classification problem (2**p**).

Solution: To find parameters that solve this classification problem, we first compute the outward-pointing normal vectors to each decision boundary (Figure 2). Denoting the point $[-4, 3]$ by A, the point $[3, 0]$ by B, and the point $[-2, -2]$ by C, the normal vectors of the decision boundaries are

$$\boldsymbol{n}_{AB} = \begin{bmatrix} 1/7 \\ 1/3 \end{bmatrix}, \quad \boldsymbol{n}_{BC} = \begin{bmatrix} 1/5 \\ -1/2\,, \end{bmatrix}, \quad \boldsymbol{n}_{CD} = \begin{bmatrix} -1/2 \\ -1/2 \end{bmatrix}. \tag{5}$$

Setting these as our weights, we obtain the weight matrix

$$\mathbb{W} = \begin{bmatrix} 1/7 & 1/3 \\ 1/5 & -1/2 \\ -1/2 & -1/5 \end{bmatrix}. \tag{6}$$

To find the thresholds θ_i, we need to solve

$$b_i = w_{i1}x_1 + w_{i2}x_2 - \theta_i = 0 \tag{7}$$

for $i = 1, \ldots, 3$, where x_1 and x_2 are coordinates on the decision boundary corresponding to index i. This equation holds because the local field b_i

changes sign at the decision boundary. We find the threshold vector

$$\boldsymbol{\theta} = \begin{bmatrix} 3/7 \\ 3/5 \\ 7/5 \end{bmatrix} . \tag{8}$$

Now consider the output weights. If we choose

$$\boldsymbol{W} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} , \tag{9}$$

a point $\boldsymbol{x}$ in the shaded region delineated by the decision boundaries gives -3, otherwise the output equals -1. Hence, if we set the output threshold Θ to a value between -3 and -1, the output is 1 outside the shaded region and -1 inside the shaded region.

4. Convolutional neural network. The two patterns shown in Figure 3 are processed by a very simple convolutional network that has one convolution layer with one single 2×2 kernel with ReLU neurons, zero threshold, and stride (1,1). The resulting feature map is fed into a 3×3 max-pooling layer with stride (1,1). Finally there is a fully connected classification layer with one output neuron with the Heaviside activation function. Determine weights of the kernel and weights and thresholds of the classification layer that allow to classify the two patterns into different classes (**2p**).

Solution: Applying the kernel to the patterns shown in Figure 3 yields

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 0 \end{bmatrix} . \tag{10}$$

Max pooling gives 1 for the left pattern, and 2 for the right pattern. Possible choice of weights for the output neuron: $W_1 = W_2 = 2$ and $\Theta = 1.5$.

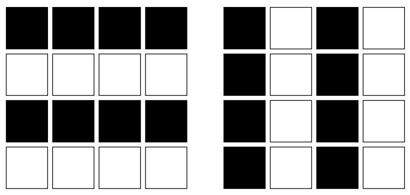

Figure 3: Patterns to be classified by convolutional network. Black squares = 1, white squares = 0. Question 4.

5. Lyapunov exponent in deep neural networks. The error in a multi-layer perceptron propagates backwards according to the rule:

$$\delta_i^{(l-1)} = \sum_j^N \delta_j^{(l)} w_{ji}^{(l)} g'(b_i^{(l-1)})\,. \tag{11}$$

Here $g' = \mathrm{d}g/\mathrm{d}b$ is the derivative of the activation function. Assume that the network is trained on random input patterns with independent bits. Further assume that the weights are random, Gaussian distributed with mean zero and variance $\langle w_{ij} w_{kl}\rangle = \sigma_w^2 \delta_{ik}\delta_{jl}$, and that the thresholds are set to zero. Here δ_{ik} is the Kronecker delta, don't confuse it with the error!

(a) Compute the mean of the error $\delta_i^{(l-1)}$ in the limit $N \to \infty$ neglecting any correlations between local fields, weights, or errors (0.5**p**).

(b) Show that the variance of the error in the limit $N \to \infty$ obeys the recursion

$$\langle (\delta_i^{(l-1)})^2\rangle = N\sigma_w^2 \langle (\delta_j^{(l)})^2\rangle \langle [g'(b_i^{(l-1)})]^2\rangle\,,$$

under the same assumptions as in task (a) (0.5**p**).

(c) It can be shown that the distribution of the local fields $b_j^{(l)}$ converges to a Gaussian with zero mean and a fixed variance σ_f^2, for large N and many layers. Assuming that the distribution has this form, derive an approximation for the maximal Lyapunov exponent. It is defined as

$$\lambda_1 = \log\left|\delta^{(l-1)}/\delta^{(l)}\right|.$$

Explain why σ_w^2 should be chosen to be on the order N^{-1} for the network to learn well. *Hint*: write $\langle [g'(b_i^{(l-1)})]^2\rangle$ as an integral expression, you do not need to evaluate the integral. (1**p**).

Solution: (a) We have $\langle \delta_i^{(l-1)}\rangle = \sum_j^N \langle \delta_j^{(l)} w_{ji}^{(l)} g'(b_i^{(l-1)})\rangle$. Neglecting correlations between local fields, weights, and errors, $\langle \delta_i^{(l-1)}\rangle = \sum_j^N \langle \delta_j^{(l)}\rangle\langle w_{ji}^{(l)}\rangle\langle g'(b_i^{(l-1)})\rangle$, we see that $\langle \delta_i^{(l)}\rangle = 0$ for $l < L$ (where L denotes the output layer).

(b) To derive the required expression, we square Eq. (11) and average. This gives

$$\begin{aligned}\langle (\delta_i^{(l-1)})^2\rangle &= \langle \sum_{j,k} \delta_j^{(l)}\delta_k^{(l)} w_{ji}^{(l)} w_{ki}^{(l)} g'(b_i^{(l-1)}) g'(b_i^{(l-1)})\rangle \\ &\approx \sum_{j,k} \langle \delta_j^{(l)}\delta_k^{(l)}\rangle\langle w_{ji}^{(l)} w_{ki}^{(l)}\rangle \langle [g'(b_i^{(l-1)})]^2\rangle\,.\end{aligned}$$

Using that $\langle w_{ji}^{(l)} w_{ki}^{(l)}\rangle = \sigma_w^2 \delta_{jk}$, we find:

$$\langle (\delta_i^{(l-1)})^2\rangle \approx N\sigma_w^2 \langle (\delta_j^{(l)})^2\rangle \langle [g'(b_i^{(l-1)})]^2\rangle\,.$$

(c) We write

$$\langle [g'(b_i^{(l-1)})]^2 \rangle = \int \mathrm{d}z P(z) [g'(z)]^2 \,,$$

where $P(z)$ is a Gaussian distribution with zero mean and variance σ_f^2. Denoting the value of the above integral by F, we find

$$\lambda_1 = \log N \sigma_w^2 F \,.$$

In order to avoid exploding or vanishing gradients, we should have $\lambda_1 \approx 0$. This is achieved by initialising the weights as $\sigma_w^2 \propto N^{-1}$. The precise constant of proportionality depends on the numerical value of F.

6. Backpropagation. To train a multi-layer perceptron by gradient descent one needs update formulae for weights and thresholds. Derive these update formulae for sequential training using backpropagation for the netweork shown in Fig. 4. The weights for the hidden layer are denoted by w_{jk}, and those for the output layer by W_{1j}. The corresponding thresholds are denoted by θ_j and Θ_1, and the activation function by $g(\ldots)$. The target values for input patterns $\boldsymbol{x}^{(\mu)}$ is $t_1^{(\mu)}$, and the pattern index μ ranges from 1 to p. The energy function is $H = \frac{1}{2}\sum_{\mu=1}^{p}(t_1^{(\mu)} - O_1^{(\mu)})^2$ (2**p**).

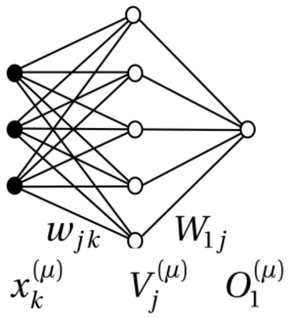

Figure 4: Multi-layer perceptron with three input terminals, one hidden layer, and one output. Question 6.

Solution: See course book.