T-61.5030 Advanced course in neural computing

Solutions for exercise 8

1. The network model is $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{n}$, where $\mathbf{y}$ is a two-dimensional vector. The goal is to analyse how much information about $\mathbf{x}$ is conserved in the output $\mathbf{y}$. This can be measured by the mutual information $I(X;Y) = h(Y) - h(Y|X)$ (where X and Y are the random variables corresponding to $\mathbf{x}$ and $\mathbf{y}$).

   Let $\mathbf{z} = \mathbf{W}\mathbf{x}$, i.e., $\mathbf{z}$ is the noiseless version of $\mathbf{y}$ and denote the variance of the components of $\mathbf{z}$ by σ_1^2 and σ_2^2. Also, denote the normalised correlation of the components z_1 and z_2 by $\rho = E\{(z_1 - E\{z_1\})(z_2 - E\{z_2\})\}/(\sigma_1\sigma_2)$ and the variance of the noise by σ_N^2 (assume the noise to be the same on both outputs).

   It is reasonable to assume the noise to be Gaussian. If the inputs are not Gaussian, neither is the noiseless output $\mathbf{z}$. However, we need to assume this for the sake of computational tractability, because then the noisy output $\mathbf{y}$ is Gaussian and we can compute $h(Y)$. Given $\mathbf{x}$ all that is unknown about $\mathbf{y}$ is the noise and therefore $h(Y|X)$ is actually the same as $h(N)$. We therefore have $I(X;Y) = h(Y) - h(Y|X) = h(Y) - h(N)$. For a certain noise level, only the term $h(Y)$ can be affected by the weights $\mathbf{W}$ and the mutual information is maximised by maximising the differential entropy $h(Y)$ of the outputs.

   For a Gaussian random variable the differential entropy is

   $$h(Y) = \frac{q}{2} + \frac{1}{2}\ln|\Sigma_Y|\,,$$

   where $q = 2$ is the dimension of $\mathbf{y}$ and $|\Sigma_Y|$ is the determinant of the covariance matrix of $\mathbf{y}$. Since

   $$\Sigma_Y = \begin{pmatrix} \sigma_1^2 + \sigma_N^2 & \sigma_1\sigma_2\rho \\ \sigma_1\sigma_2\rho & \sigma_2^2 + \sigma_N^2 \end{pmatrix}$$

   we have

   $$|\Sigma_Y| = (\sigma_1^2 + \sigma_N^2)(\sigma_2^2 + \sigma_N^2) - (\sigma_1\sigma_2\rho)^2 = \sigma_N^4 + \sigma_N^2(\sigma_1^2 + \sigma_2^2) + \sigma_1^2\sigma_2^2(1 - \rho^2)\,.$$

   If the noise level is high compared to the noiseless output $\mathbf{z}$ ($\sigma_N^2 \gg \sigma_1^2$, $\sigma_N^2 \gg \sigma_2^2$), the last term $\sigma_1^2\sigma_2^2(1-\rho^2)$ is negligible compared to $\sigma_N^2(\sigma_1^2 + \sigma_2^2)$. This means that the network tries to maximise the variances of the outputs y_1 and y_2 without paying any attention to their correlation which means the outputs should probably be the same, i.e., the one that has the highest variance.

   If the noise level is low compared to the noiseless output, the middle term $\sigma_N^2(\sigma_1^2 + \sigma_2^2)$ is negligible and the network tries to maximise $\sigma_1^2\sigma_2^2(1-\rho^2)$. The variances of the outputs should be high, but the correlation of the outputs low. This means that the network represents two uncorrelated aspects of the inputs.

2. The goal of the Infomax learning rule is to conserve as much information about the input x in the output y, i.e., to maximise the mutual information $I(X;Y) = h(Y) - h(Y|X)$. If there is noise in the inputs, this reduces into $h(Y) - h(N)$ as was argued in the previous

exercise. In this problem there is no noise, which can be seen as the limiting case $\sigma_N \to 0$. This makes the term $h(N)$ approach $-\infty$, but this doesn't matter since the weights can only affect the term $h(Y)$ and therefore only that term matters. Maximising the mutual information is thus equivalent to maximising the differential entropy $h(Y)$ of the output. The output $g(x)$ is restricted to the range $[0, 1]$ which means that the maximum would be reached if the output had a uniform distribution between zero and one.

Recall that the relation $y = g(x)$ means that the density of y is

$$p(y) = \frac{p(x)}{|g'(x)|} .$$

We then have

$$h(Y) = -\int p(y) \ln p(y) dy = -\int p(y) \ln p(x) dy + \int p(y) \ln |g'(x)| dy =$$

$$-\int p(x) \ln p(x) dx + E\left\{\ln |g'(x)|\right\} .$$

The first term does not depend on the weights of the network and therefore only the second term needs to be taken into account in the learning.

If the learning is done by stochastic gradient descent we have

$$\Delta w \propto \frac{\partial}{\partial w} h(y) = \frac{\partial}{\partial w} ln|g'(x)| = \frac{\frac{\partial}{\partial w} g'(x)}{g'(x)} = \frac{\frac{\partial}{\partial w} y(1-y)w}{y(1-y)w}$$

because the function $y = f(u) = 1/(1 + e^{-u})$ has the nice property $f'(u) = y(1 - y)$ and if we denote $u = wx + w_0$, we have $y = f(u)$ and $g'(x) = f'(u)\partial u/\partial x$ by the chain rule. Likewise, $\partial y/\partial w = f'(u)\partial u/\partial w = y(1 - y)x$, which means that

$$\Delta w \propto \frac{(1-2y)y(1-y)xw + y(1-y)}{y(1-y)w} = \frac{1}{w} + x(1-2y) .$$

A similar derivation yields

$$\Delta w_0 \propto 1 - 2y .$$

In the multivariate case the scalar weight w is replaced by a matrix $\mathbf{W}$ and a similar derivation would give the learning algorithm in equation (10.137) of Haykin's book.

3. By definition, we have

$$p_{ij}^{(n)} = P(x_t = j | x_{t-n} = i)$$

where t denotes time and n denotes the number of discrete steps. For $n = 1$ we have the one-step transition probability

$$p_{ij}^{(1)} = p_{ij} = P(x_t = j | x_{t-1} = i) .$$

For $n = 2$ we have the two-step transition probability

$$p_{ij}^{(2)} = \sum_k p_{ik} p_{kj}$$

where the sum is taken over all intermediate steps k taken by the system. By induction, it thus follows that

$$p_{ij}^{(n+1)} = \sum_k p_{ik} p_{kj}^{(n)} .$$

4. The stochastic matrix of the Markov chain in Haykin, Fig. P.11.4 is given by

$$P = \begin{bmatrix} \frac{3}{4} & \frac{1}{4} & 0 \\ 0 & \frac{2}{3} & \frac{1}{3} \\ \frac{1}{4} & 0 & \frac{3}{4} \end{bmatrix}$$

Let π_1, π_2, and π_3 denote the steady-state probabilities of this chain. We may then write (see Haykin, Eq. (11.27))

$$\begin{aligned} \pi_1 &= \tfrac{3}{4}\pi_1 + 0\pi_2 + \tfrac{1}{4}\pi_3 \ , \\ \pi_2 &= \tfrac{1}{4}\pi_1 + \tfrac{2}{3}\pi_2 + 0\pi_3 \ , \text{ and} \\ \pi_3 &= 0\pi_1 + \tfrac{1}{3}\pi_2 + \tfrac{3}{4}\pi_3 \ . \end{aligned}$$

The first and third equations give

$$\pi_1 = \pi_3 \text{ and } \pi_2 = \frac{3}{4}\pi_3 \ .$$

We also have, by definition, $\pi_1 + \pi_2 + \pi_3 = 1$. Hence,

$$\pi_3 + \frac{3}{4}\pi_3 + \pi_3 = 1$$

or equivalently $\pi_3 = \frac{4}{11}$, and so

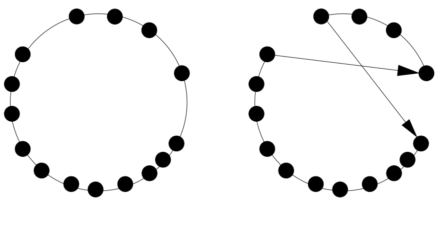
$$\pi_1 = \frac{4}{11} \text{ and } \pi_2 = \frac{3}{11} \ .$$

5. Simulated annealing algorithm for solving the travelling salesman problem:

   (a) Set up an annealing schedule for the algorithm.

   (b) Initialize the algorithm by picking a tour at random.

   (c) Choose a pair of cities in the tour and reverse the order that the cities in-between the selected pair are visited. This procedure, illustrated below, generates new tours in a local manner:



   (d) Calculate the energy difference due to the reversal of paths applied in step (c).

   (e) If the energy difference so calculated is negative or zero, accept the new tour. If, on the other hand, it is positive, accept the change in the tour with probability defined in accordance with the Metropolis algorithm.

   (f) Select another pair of cities, and repeat steps (c) to (e) until the required number of iterations is complete.

   (g) Lower the temperature in the annealing schedule, and repeat steps (c) to (f).