Solutions to the exam in Neural Networks and Learning Systems - TBMI26 Exam 2019-08-31

Part 1

1. examples:
   Supervised learning: Back-propagation, Support vector machines, k-NN,...
   Unsupervised learning: Prinicpal component analysis, Independent component analysis, canonical correlation analysis, self-organizing maps,...
   Reinforcement learning: Q-learning, genetic algorithms, ...
2. The input value to the bias weight does not matter, as long as it is not zero!
3. 1-C, 2-A, 3-B
4. In k-means clustering, only the centers of the clusters are modelled, whereas in Mixture of Gaussian clustering, also the cluster shape is modelled using the covariance matrix, which can model circular and elliptical cluster shapes.
5. d is the distance between the cluster centers and v is the variance of the projected clusters.
6. ϵ-*greedy exploration* means that a random action (exploration) is made with probability ϵ and a greedy action (exploitation), i.e., an action that maximizes the reward, is made with probability $1 - \epsilon$.
7. The *momentum* term.
8. A linear activation function would give the result according to the dashed line while a sigmoid activation function would give a result similar to the solid line.
9. ± 1
10. Convolutional neural networks (CNN).

Part 2

11. It refers to the delayed feedback, i.e. that you do not know which action is responsible for the reward.

12. In the case of $k = 1$ X will belong to a black dot, since this is the closest sample.

    When $k = 3$ X will be classified as a square because two of the three closest samples are squares.

    In the case of $k = 2$ the votes is even between the two classes and some confusion occurs. One solution is to classify the X as a dot since this is the closest sample.

13. A kernel function defines the inner product $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_2)$ in the new feature space. Thus, $\kappa(\mathbf{x}_1, \mathbf{x}_2)$ specifies the feature space by defining how distances and angles are measured, instead of explicitly stating the mapping function $\Phi(\mathbf{x})$.

    The distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ in the new feature space is

$$\begin{aligned}
\|\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2)\| &= \sqrt{(\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2))^T (\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2))} \\
&= \sqrt{\Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_1) - 2\Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_2) + \Phi(\mathbf{x}_2)^T \Phi(\mathbf{x}_2)} \\
&= \sqrt{\kappa(\mathbf{x}_1, \mathbf{x}_1) - 2\,\kappa(\mathbf{x}_1, \mathbf{x}_2) + \kappa(\mathbf{x}_2, \mathbf{x}_2)} \\
&= \sqrt{5 - 4 + 2} = \sqrt{3}
\end{aligned}$$

14. The data is divided into a training set, a validation set and test set. the training set is used for updating the parameters. The validation set is used for checking the generalization error during training. The test set is used after trining for evaluating the final performance.

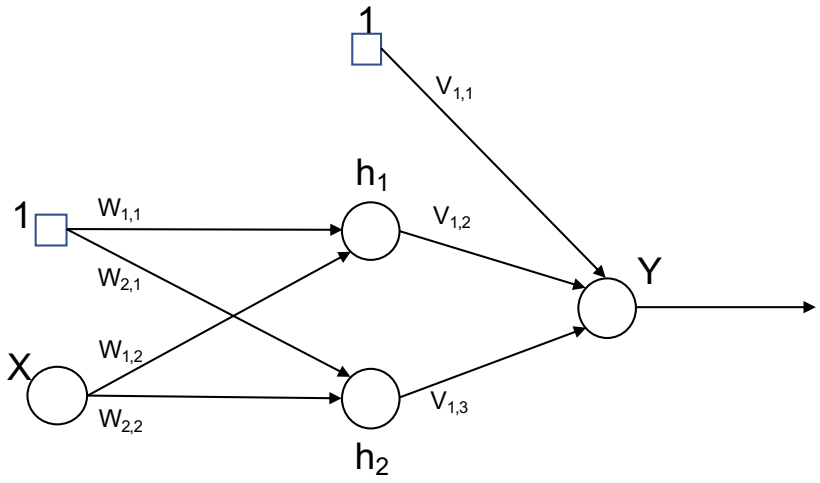
15. A decision stump function $f(x)$ is simply a threshold operation $f(x) = sign(x - \tau)$, i.e., it deterimines a binary class label as

$$f(x) = \begin{cases} 1 & \text{if } x \geq \tau \\ -1 & \text{if } x < \tau \end{cases}$$

    One may also have to reverse the polarity so that $f(x) = -sign(x - \tau)$. $f(x)$ is not differentiable, so it must be trained using a brute force method which means that the threshold τ is selected by trying a large number of values and selecting the value that gives the minimum classification error.

Part 3

16. a) One possible solution is the network illustrated below.



where

$$\mathbf{W} = \begin{bmatrix} 0.5 & 1 \\ 0.3 & -1 \end{bmatrix}$$
$$\mathbf{V} = \begin{bmatrix} -1 & 1 & 1 \end{bmatrix}$$

After forward propagation through the network with *sign* as activation function in the hidden layer and in the output layer, Y will be:

$$\mathbf{Y} = \begin{bmatrix} -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 \end{bmatrix}$$

$\mathbf{Y} = \mathbf{D}$, giving an accuracy of 100 %.

b) The weights should be initialized randomly. The net is forward propagated and the error (using a suitable error function) is calculated. The minimum error is found using gradient search: $\frac{\delta\epsilon}{\delta\mathbf{V}}$ and $\frac{\delta\epsilon}{\delta\mathbf{W}}$. The weights are updated using according to: $\mathbf{V_{i+1}} = \mathbf{V_i} - \eta\frac{\delta\epsilon}{\delta\mathbf{V}}$ and $\mathbf{W_{i+1}} = \mathbf{W_i} - \eta\frac{\delta\epsilon}{\delta\mathbf{W}}$. A different more suitable activation function, e.g. $tanh()$, must be used since *sign* is not differentiable.

17. (One way to do this is presented in the lectures. One alternative is given below.)

The first principal direction $\mathbf{w} = (w_1, \ldots, w_n)^T$ is the direction that maximizes the variance $V\left(\mathbf{w}^T\mathbf{x}\right)$. If we first remove the average from each variable in $\mathbf{x}$, then the variance is given by

$$V\left(\mathbf{w}^T\mathbf{x}\right) = E\left(\left(\mathbf{w}^T\mathbf{x}\right)^2\right) = E\left(\left(\mathbf{w}^T\mathbf{x}\right)\left(\mathbf{x}^T\mathbf{w}\right)\right) = \mathbf{w}^T E\left(\mathbf{x}\mathbf{x}^T\right)\mathbf{w} = \mathbf{w}^T\mathbf{C}\mathbf{w}$$

A simple way to maximize $\mathbf{w}^T\mathbf{C}\mathbf{w}$ is to scale $\mathbf{w}$. We therefore have to limit the length of $\mathbf{w}$ so that only the direction is considered. The problem then becomes

$$max \mathbf{w}^T\mathbf{C}\mathbf{w}$$
$$\text{with constraint} \|\mathbf{w}\|^2 = 1$$

We include the constraint into the target function using a Lagrange-parameter λ:

$$max \mathbf{w}^T\mathbf{C}\mathbf{w} + \lambda\left(\|\mathbf{w}\|^2 - 1\right)$$

The derivative of the expression above with respect to $\mathbf{w}$ is

$$2\mathbf{C}\mathbf{w} + 2\lambda\mathbf{w},$$

and if we set it to 0 we get the equation

$$\mathbf{C}\mathbf{w} = \lambda\mathbf{w}$$

The $\mathbf{w}$ we want to find is therefore an eigenvector to the covariance matrix $\mathbf{C}$. If we denote the eigenvectors of $\mathbf{C}$ as $\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_n$ and use the eigenvector $\hat{\mathbf{w}}_i$ we get the variance

$$\hat{\mathbf{w}}_i^T \underbrace{\mathbf{C}\hat{\mathbf{w}}_i}_{\lambda_i\hat{\mathbf{w}}_i} = \lambda_i\hat{\mathbf{w}}_i^T\hat{\mathbf{w}}_i = \lambda_i,$$

i.e. the eigenvalues give the variance in the direction of the corresponding eigenvector. The solution is consequently given by the eigenvector that corresponds to the largest eigenvalue.

18. Initially, we have the dataset and the two prototypes (k=2) as presented in figure 1.

a) k-Means has two alternating phases, *sample assignment* and prototype update. During the first phase, we assign each data sample to the closest (Euclidean) prototype. In the second phase, we update the prototypes using the assigned samples: $\mathbf{p}_j = \frac{1}{|S_j|}\sum_{k\in S_j}\mathbf{x}_k$, where S_j is the set of data samples belonging to prototype j. In the first iteration, the left hand samples are closest to $\mathbf{p}_1$ and the right hand samples are closest to $\mathbf{p}_2$. We then update the prototypes to $\mathbf{p}_1 = \begin{bmatrix}-1\\0\end{bmatrix}$ and $\mathbf{p}_2 = \begin{bmatrix}+1\\0\end{bmatrix}$. During the second iteration, the assigned sets do not change. This implies no change in the prototypes either. The k-Means method has converged.

b) Mixture of Gaussians (MoG) has the same alternating phases, *sample assignment* and *prototype update*. The 'closeness' measure during the first phase is, however, here replaced with finding the Gaussian prototype which gives the highest likelihood for each sample $\mathbf{x}_i$, i.e. finding the prototype j which maximizes

$$p(\mathbf{x}_i;\mathbf{m}_j,\mathbf{C}_j) = \frac{1}{(2\pi)^{N/2}\sqrt{\det\mathbf{C}_j}}\exp\left[-\frac{1}{2}(\mathbf{x}_i-\mathbf{m}_j)^T\mathbf{C}_j^{-1}(\mathbf{x}_i-\mathbf{m}_j)\right]$$

Since $\mathbf{C}_1 = \mathbf{C}_2$ and they are isotropic ($= \mathbf{I}$), it is enough to compare the euclidean distance from the sample to the prototype, i.e. we have the same situation as in a): the left hand samples are closest to $\mathbf{p}_1$ and the right hand samples are closest to $\mathbf{p}_2$. We then update the prototypes to $\mathbf{p}_1 = \begin{bmatrix}-1\\0\end{bmatrix}$ and $\mathbf{p}_2 = \begin{bmatrix}+1\\0\end{bmatrix}$. The covariance matrices are updated as $\mathbf{C}_j = \frac{1}{|S_j|}\sum_{k\in S_j}(\mathbf{x}_k-\mathbf{p}_j)(\mathbf{x}_k-\mathbf{p}_j)^T$, where S_j is the set of data samples assigned to prototype j as before. We get $\mathbf{C}_1 = \mathbf{C}_2 = \frac{1}{3}\begin{bmatrix}0&0\\0&2\end{bmatrix}$. This corresponds to Gaussians which have no extension/width/variance in the first dimension, a variance of $\frac{2}{3}$ in the second dimension and no covariance between the dimensions $\rightarrow$ 'a very thin vertical ellipse' as a general shape.

19. A definition of an optimal (deterministic) Q-function:

$$\begin{aligned} Q^*(x,a) &= r(x,a) + \gamma V^*(f(x,a)) && (1)\\ &= r(x,a) + \gamma Q^*(f(x,a), \mu^*(f(x,a))) && (2)\\ &= r(x,a) + \gamma \max_b Q^*(f(x,a),b) && (3) \end{aligned}$$

and for the V-function:

$$V^*(x) = \max_b Q^*(x,b) \quad (4)$$

We can find a solution by going through the state models recursively.
System A:

$$\begin{aligned} V^*(3) &= 1 + \gamma V^*(2) && (5)\\ V^*(2) &= 3 + \gamma V^*(3) && (6)\\ & && (7) \end{aligned}$$

Combining these two equations yields the following:

$$\begin{aligned} V^*(3) &= 1 + \gamma\left(3 + \gamma V^*(3)\right) \implies && (8)\\ V^*(3) &= \frac{1+3\gamma}{1-\gamma^2} && (9)\\ V^*(2) &= \frac{3+\gamma}{1-\gamma^2} && (10)\\ V^*(1) &= 1 + \gamma V^*(2) = \frac{1+3\gamma}{1-\gamma^2} && (11) \end{aligned}$$

Since there are no branching paths in this system, the Q-function is equal to the V-function:

$$\begin{aligned} Q^*(1,up) &= V^*(1) && (12)\\ Q^*(2,up) &= V^*(2) && (13)\\ Q^*(3,down) &= V^*(3) && (14)\\ & && (15) \end{aligned}$$

System B:

$$\begin{aligned} V^*(4) &= 0 && (16)\\ V^*(3) = Q(3,up) &= 4 + \gamma V^*(4) = 4 && (17)\\ Q^*(2,left) &= 1 + \gamma V^*(4) = 1 && (18)\\ Q^*(2,up) &= 0 + \gamma V^*(3) = 4\gamma && (19)\\ & && (20) \end{aligned}$$

The action "left" in state 2 will be taken if:

$$\begin{aligned} Q^*(2,up) &< Q^*(2,left) \implies && (21)\\ 4\gamma &< 1 \implies && (22)\\ \gamma &< \frac{1}{4} && (23) \end{aligned}$$

Thus:

$$V^*(2) = \begin{cases} 1 & \text{if } \gamma \leq 0.25 \\ 4\gamma & \text{if } \gamma > 0.25 \end{cases} \tag{24}$$

For state 1:

$$Q^*(1, right) = 1 + \gamma V^*(3) = 1 + 4\gamma \tag{25}$$

$$Q^*(1, up) = 2 + \gamma V^*(2) = \begin{cases} 2 + \gamma & \text{if } \gamma \leq 0.25 \\ 2 + 4\gamma^2 & \text{if } \gamma > 0.25 \end{cases} \tag{26}$$

If $\gamma \leq 0.25$ then action "up" will be taken in state 1 if:

$$Q^*(1, right) < Q^*(1, up) \implies \tag{27}$$

$$1 + 4\gamma < 2 + \gamma \implies \tag{28}$$

$$\gamma < \frac{1}{3} \tag{29}$$

Since this follows from the assumption that $\gamma \leq 0.25$ this is always true, so for $\gamma \leq 0.25$ action "up" will always be chosen in state 1. For $\gamma > 0.25$ the Q-function for "up" and "right" is equal when

$$Q^*(1, right) = Q^*(1, up) \implies \tag{30}$$

$$1 + 4\gamma = 2 + 4\gamma^2 \implies \tag{31}$$

$$4\gamma^2 - 4\gamma + 1 = 0 \implies \tag{32}$$

$$\gamma = \frac{1}{2} \text{ (double root)} \tag{33}$$

Since this is a double root either $Q^*(1, up)$ or $Q^*(1, right)$ will always be greater, only equal to each other in $\gamma = 0.5$. Test one value:

$$Q^*(1, right)|_{\gamma=1} = 5 \tag{34}$$

$$Q^*(1, up)|_{\gamma=1} = 6 \tag{35}$$

Thus $Q^*(1, up) \geq Q^*(1, right)$ for all γ, and the action "up" will always be chosen. We can now set:

$$V^*(1) = Q^*(1, up) = \begin{cases} 2 + \gamma & \text{if } \gamma \leq 0.25 \\ 2 + 4\gamma^2 & \text{if } \gamma > 0.25 \end{cases} \tag{36}$$

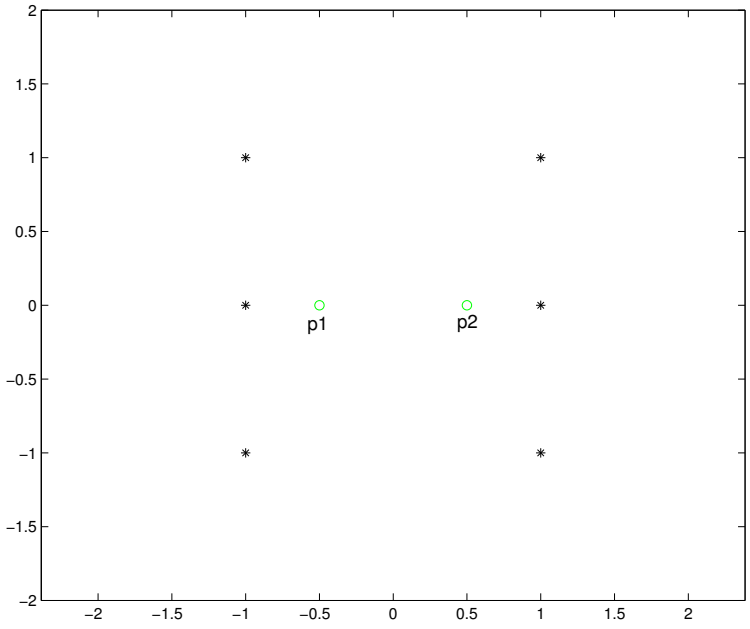


Figure 1: Initial data set and prototypes (problem 18)