# Solutions to the exam in
# Neural Networks and Learning Systems - TBMI26
# Exam 2019-06-12

Part 1

1. - k-Nearest Neighbors - (S)
   - Support Vector Machines - (S)
   - AdaBoost - (S)
   - Principal Component Analysis (U)
   - Multi-layer Perceptron (Neural Network) (S)
   - Mixture of Gaussian Clustering (U)
2. The accuracy is usually calculated as the number of correctly classified examples divided with the total number of training examples.
3. The hidden layers require non-linear differentiable activations which non of these are.
4. The first eigenvecor of the data covariance matrix shows the direction in which the data have maximum variance.
5. That both classes have the same covariance matrix, i.e. that the shapes of the distributions are identical.
6. The value of the V function for is the maximum Q-value over all possible actions.
7. To speed up the convergence
8. In regression tasks, in particular when the output needs to be outside the interval -1 to 1.
9. Slack-variables are used to allow som samples to be miss-classified, in order not to overfit to noisy data and outliers.
10. The wight matrix in a convolutional layer is sparse (and with constant diagonals i.e. a Toeplitz matrix)

Part 2

11. The two (main) differences between supervised and reinforcement learning are that in RL, the feedback is delayed, and that now ground truth (desired output) is available in RL.

12. The first step is to cluster all points to belong to the closest prototype. The second step is to compute new prototypes as the mean vector of each cluster.

13.
    - $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\varphi}(\mathbf{x}_i), \boldsymbol{\varphi}(\mathbf{x}_j) \rangle$, i.e. it implicitly calculates the standard scalar product in feature space, expressed in the input space.
    - $\kappa(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x})^T \varphi(\mathbf{y}) = x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 x_2 y_1 y_2 = (x_1 y_1 + x_2 y_2)^2 = \left(\mathbf{x}^\mathbf{T} \mathbf{y}\right)^2$

14. ReLU = "Rectified Linear Unit", which means that y = max(s,0).
    a) The ReLU does not suffer from the "vanishing gradient" problem in deep networks. It is also computationally efficient. b) The "knockout problem": A neuron can be driven to a state where it never activates for any input.

15. $\|\mathbf{w}\|$ is minimized and the equation holds with equality for the *support vectors*.

Part 3

16. a)

$$\mathrm{C} = \begin{array}{|c|c|c|c|}\hline \mathbf{2} & 1 & 5 & 2 \\ \hline 1 & 7 & 1 & 6 \\ \hline 4 & 2 & 6 & 0 \\ \hline \end{array}$$

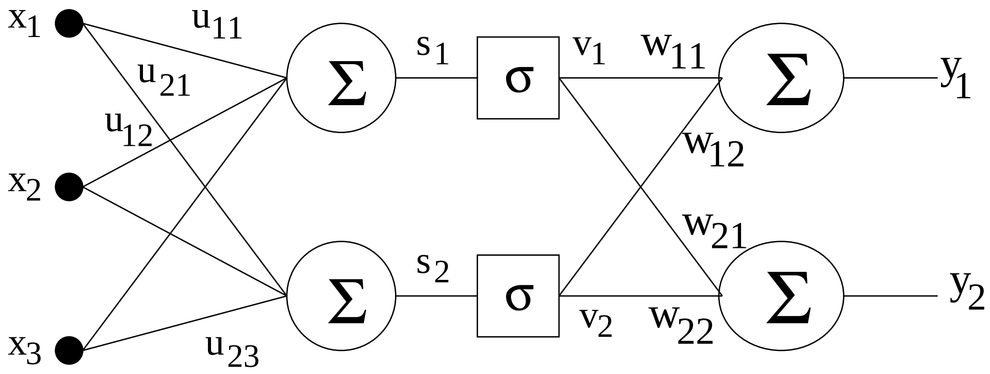
b)

$$\mathrm{AA} = \begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & \mathbf{0} & 1 & 0 & 2 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 2 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

c) Image width (or height) after first convolution: $512-3+1 = 510$. The activation does not change the size. After pooling: $\lceil 510/2 \rceil = 255 = 2^8 - 1$. After second correlation: $255 - 3 + 1 = 253$. After pooling: $\lceil 253/2 \rceil = 127 = 2^7 - 1$. The image size becomes $1 = 2^1 - 1$ after $N = 8$ complex layers.

17. a)

The network look like this



The updates are

$$\Delta w_{ji} = -\eta \frac{\partial \varepsilon}{\partial w_{ji}}$$

where $\varepsilon$ is defined as

$$\varepsilon = \frac{1}{2} \sum_k e_k^2 = \frac{1}{2} \sum_k (d_k - y_k)^2$$

For the output layer we have

$$\frac{\partial \varepsilon}{\partial w_{ji}} = \frac{\partial \varepsilon}{\partial e_j} \frac{\partial e_j}{\partial y_j} \frac{\partial y_j}{\partial w_{ji}} = -e_j v_i = -e_j \sum_k x_k u_{ik}$$

and for the hidden layer we have

$$\frac{\partial \varepsilon}{\partial u_{ji}} = \sum_k \frac{\partial \varepsilon}{\partial e_k} \frac{\partial e_k}{\partial y_k} \frac{\partial y_k}{\partial v_j} \frac{\partial v_j}{\partial s_j} \frac{\partial s_j}{\partial u_{ji}} = -\sum_k e_k w_{jk} x_i$$

The updates in the output layer become:

$$\Delta w_{ji} = \eta e_j \sum_k x_k u_{ik}$$

...and in the hidden layer:

$$\Delta u_{ji} = \eta \sum_k e_k w_{jk} x_i$$

b) The lack of bias weights restricts the decision boundaries so they have to go through the origin.

18. We had the following data:

$$\mathbf{X}_{Train} = \begin{bmatrix} 1 & 2 & 2 & 2 & 3 & 4 \\ 1 & 1 & 2 & 3 & 3 & 3 \end{bmatrix} \qquad \mathbf{y}_{Train} = \begin{bmatrix} 1 & -1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

$$\mathbf{X}_{Test} = \begin{bmatrix} 0 & 1 & 2 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 & 3 & 3 \end{bmatrix} \qquad \mathbf{y}_{Test} = \begin{bmatrix} 1 & -1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

where $\mathbf{X}_{Train}$ & $\mathbf{X}_{Test}$ contain six 2d-samples (one per column), and $\mathbf{y}_{Train}$ & $\mathbf{y}_{Test}$ contain classification labels for the corresponding samples.

a) We have performed the third iteration of AdaBoost using 'decision stumps' as weak classifiers. We updated the weights (d), classification labels (c) and $\alpha$ for each weak classifier accordingly:

$$\mathbf{d} = \begin{bmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/4 & 1/8 & 1/8 & 1/8 & 1/8 & 1/4 \\ 1/6 & 1/12 & 1/12 & 1/4 & 1/4 & 1/6 \\ 1/10 & 1/4 & 1/4 & 3/20 & 3/20 & 1/10 \end{bmatrix}$$

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

$$\alpha = \begin{bmatrix} 0.35 \\ 0.55 \\ 0.80 \end{bmatrix}$$

b) The accuraccy for $\mathbf{X}_{Train} = 1$ and for $\mathbf{X}_{Test} = \frac{5}{6}$

c) It is not possible. We have two overlapping data observations at position $[1, 1]$ in the test and training data set but they belong to different classes.

19. The estimated Q-function is defined by: $\hat{Q}(s_k,a_j) \leftarrow (1-\alpha)\hat{Q}(s_k,a_j)+\alpha\Big(r+\gamma\max_a\hat{Q}(s_{k+1},a)\Big)$

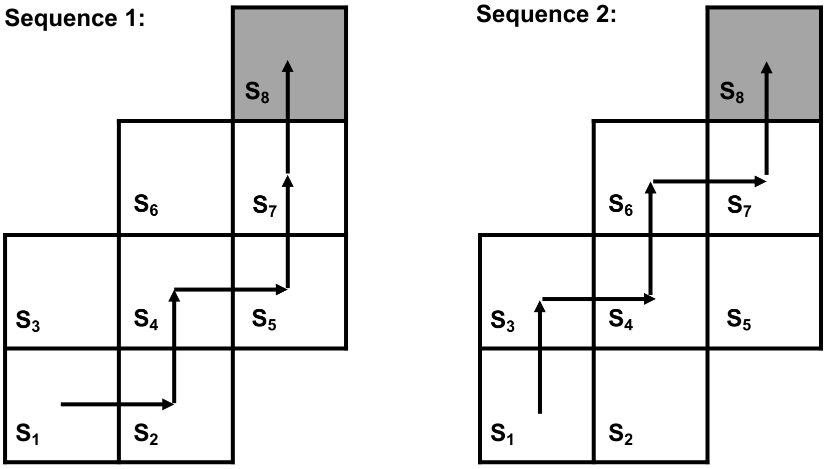


Figure 1: Sequences of action

After the first sequence, the only state that is affected is $s_7$:

$$\hat{Q}(s_7,up) = (1-\alpha)\cdot 0 + \alpha(5+\gamma\cdot 0) = 5\alpha$$

The updated Q-function will be:

| **Q(s,a)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| **Right** | 0 | - | 0 | 0 | - | 0 | - | *end* |
| **Up** | 0 | 0 | - | 0 | 0 | - | $5\alpha$ | *end* |

After the second sequence, the following states are updated accordingly:

$$\begin{aligned}\hat{Q}(s_4,up) &= (1-\alpha)\cdot 0 + \alpha(-1+\gamma\cdot 0) = -\alpha\\ \hat{Q}(s_6,right) &= (1-\alpha)\cdot 0 + \alpha(0+\gamma\cdot 5\alpha) = 5\alpha^2\gamma\\ \hat{Q}(s_7,up) &= (1-\alpha)\cdot 5\alpha + \alpha(5+\gamma\cdot 0) = 5\alpha(2-\alpha)\end{aligned}$$

The updated Q-function will be:

| **Q(s,a)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| **Right** | 0 | - | 0 | 0 | - | $5\alpha^2\gamma$ | - | *end* |
| **Up** | 0 | 0 | - | $-\alpha$ | 0 | - | $5\alpha(2-\alpha)$ | *end* |

After the third sequence (sequence 1 again), the following states are updated accordingly:

$$\begin{aligned}\hat{Q}(s_5,up) &= (1-\alpha)\cdot 0 + \alpha(0+\gamma\cdot 5\alpha(2-\alpha)) = 5\alpha^2\gamma(2-\alpha)\\ \hat{Q}(s_7,up) &= (1-\alpha)\cdot 5\alpha(2-\alpha) + \alpha(5+\gamma\cdot 0) = 5\alpha(3-3\alpha+\alpha^2)\end{aligned}$$

The updated Q-function will be:

| **Q(s,a)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| **Right** | 0 | - | 0 | $-\alpha$ | - | $5\alpha^2\gamma$ | - | *end* |
| **Up** | 0 | 0 | - | 0 | $5\alpha^2\gamma(2-\alpha)$ | - | $5\alpha(3-3\alpha+\alpha^2)$ | *end* |