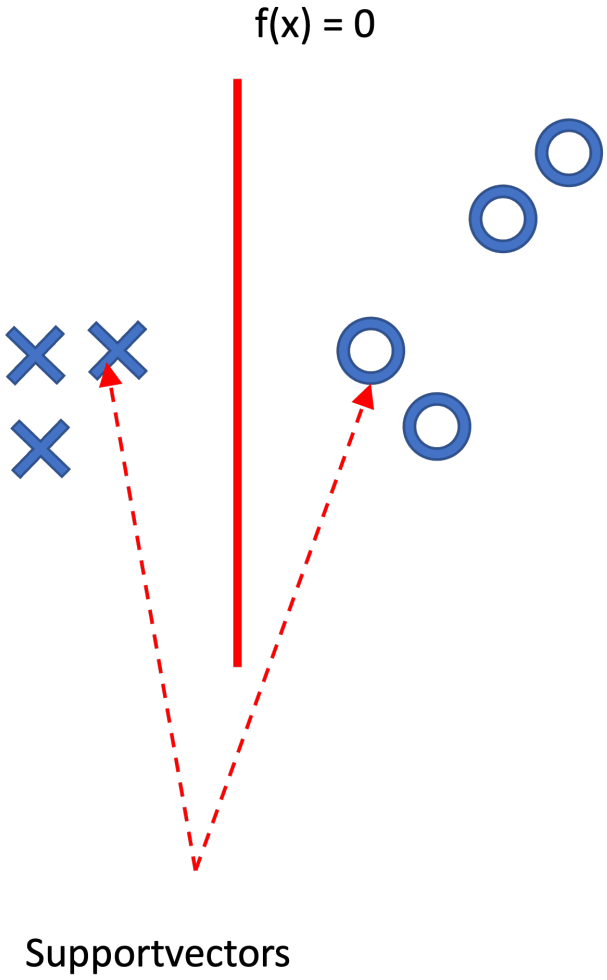
Solutions to the exam in
Neural Networks and Learning Systems - TBMI26 / 732A55
Exam 2023-03-20

Part 1

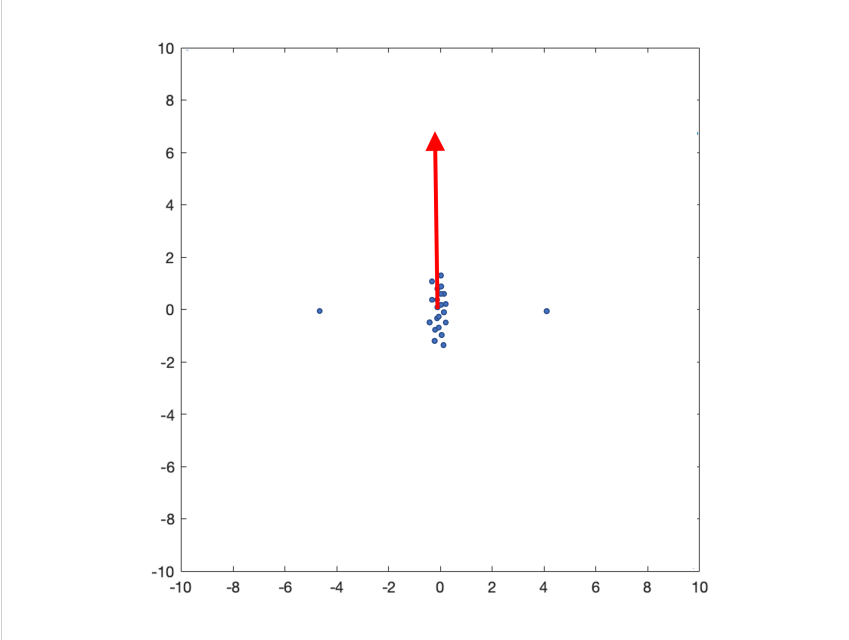
1.
   - k-means - Unsupervised learning
   - Q-learning - Reinforcement learning
   - Ada-Boost - Supervised learning

2. See figure below



3. Outliers receive more and more weight because they are frequently misclassified, so that eventually the training is only focussed on classifying the outliers correctly.

4. See figure below.



5. The consequence of overfitting is that you get poor generalisation when using the model on new data.

6. The value of k should increase.

7. The purpose of data augmentation is to avoid overfitting by artificially increasing the number of training data examples, e.g. by rotation and translation of the data.

8. A high value of the discount factor will lead to a behavior that focuses on long-term rewards, whereas a low value will focus the system on short-term rewards.

9. $k(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} e^{-(x_i^2+y_i^2)}$

10. The accuracy is the number of correctly classified examples divided with the total number of samples, i.e., (80+90)/(80+90+20+30) = 0.77.

11. The U-net is a suitable architecture for image segmentation. See the lecture for an illustration.

12. The reason is the "vanishing gradient" problem which is caused by the repeated multiplication of gradients which are small almost everywhere for the sigmoid function. To avoid this, we can us alternative activation functions such as the ReLU function.

13. For example this:

-1	-1	1
0	0	0
1	1	1

14. The empirical risk/0-1 loss function is defined as

$$\epsilon(\mathbf{w}) = \sum_{i=1}^{N} f(\mathbf{x}_i; \mathbf{w}) \neq y_i,$$

i.e., the number of training data examples that are falsely classified. This function is not differentiable so that one must resort to brute force optimization and systematically or randomly testing different choices of parameters $\mathbf{w}$. This is for example done when training decision stumps in the AdaBoost algorithm.

15. The two main building blocks in Generative Adversarial Networks are the Generator and the Discriminator. The task for the generator is to generate synthesized data that is similar to reala data and the task for the discriminator is to discriminate between real and synthetic data.

Part 2

1. a) The minimal version of the model which can classify the dataset should include the *tanh* activation in the hidden layer (0.5p) and a bias in both the hidden and output layers (0.5p). Figure **??** shows a labelled schematic of the network with all the variables and their dimensions included. The activation in the output layer is optional. The activation in the hidden layer is needed given that the dataset is not linearly separable. The bias is needed since the feature space of the input is not zero-centred.

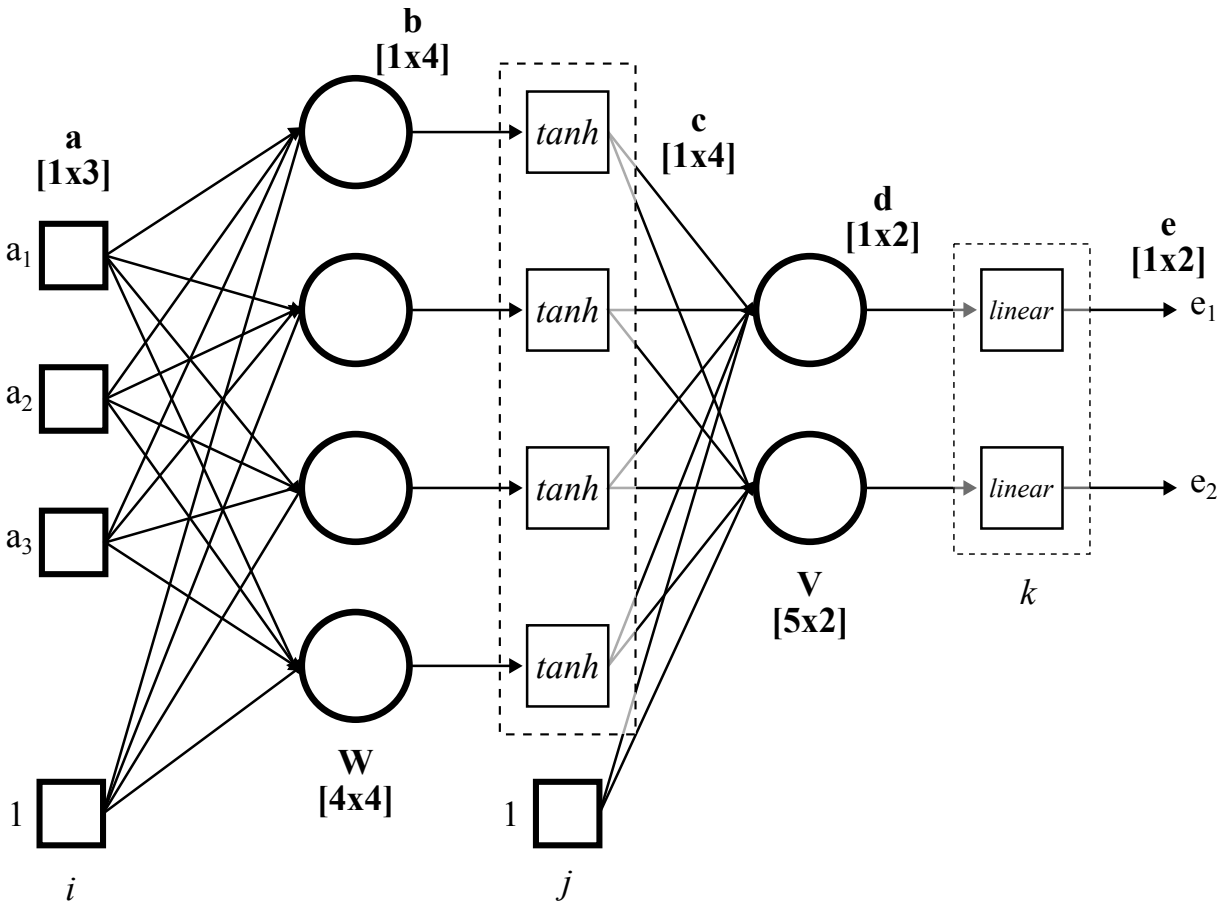


Figure 1: Schematic of a two layer fully connected network.

b) All the variables should be added as in Figure **??** along with their dimensions (0.5p). The steps of the forward pass are:

$$a_i = i-\text{th input with the added bias}, \quad b_j = \sum_{i=1}^{4} a_i w_{i,j},$$

$$c_j = \begin{cases} tanh(b_j) & j = \{1,2,3,4\}, \\ 1, & j = 5 \end{cases} \qquad d_k = \sum_{j=1}^{5} c_j v_{j,k}, \qquad e_k = (\text{no activation}) = d_k$$

$$\epsilon = \sum_{k=1}^{2} (e_k - g_k)^2$$

where g_k is the target value for the k-th output (0.5p).
From the above definitions, we calculate all partial derivatives between contiguous stages of the network (1p):

$$\begin{aligned}
\frac{\partial \epsilon}{\partial e_k} &= \frac{\partial}{\partial e_k}\sum_{k=1}^{2}(e_k - g_k)^2 = 2(e_k - g_k),\\
\frac{\partial e_k}{\partial d_k} &= \frac{\partial}{\partial d_k} d_k = 1,\\
\frac{\partial d_k}{\partial c_j} &= \frac{\partial}{\partial c_j}\sum_{j=1}^{5} c_j v_{j,k} = v_{j,k}, \quad \frac{\partial d_k}{\partial v_{j,k}} = \frac{\partial}{\partial v_{j,k}}\sum_{j=1}^{5} c_j v_{j,k} = c_j,\\
\frac{\partial c_j}{\partial b_j} &= \frac{\partial}{\partial b_j} = (1 - tanh(b_j)^2) = (1 - c_j^2) \quad j = 1,2,3,4\\
\frac{\partial b_j}{\partial w_{i,j}} &= \frac{\partial}{\partial w_{i,j}}\sum_{i=1}^{4} a_i w_{i,j} = a_i
\end{aligned}$$

Then apply the chain rule:

$$\frac{\partial \epsilon}{\partial v_{j,k}} = \frac{\partial \epsilon}{\partial e_k}\frac{\partial e_k}{\partial d_k}\frac{\partial d_k}{\partial v_{j,k}} = 2(e_k - g_k)c_j \quad (1p)$$

and

$$\frac{\partial \epsilon}{\partial w_{i,j}} = \left(\sum_{k=1}^{2}\frac{\partial e_k}{\partial d_k}\frac{\partial d_k}{\partial c_j}\right)\frac{\partial c_j}{\partial b_j}\frac{\partial b_j}{\partial w_{i,j}} = 2\left(\sum_{k=1}^{2}(e_k - y_k)v_{j,k}\right)(1 - c_j^2)a_i \quad (1p)$$

and the weight update rules are

$$\begin{aligned}
v_{j,k} &\leftarrow v_{j,k} - \eta\frac{\partial \epsilon}{\partial v_{j,k}},\\
w_{i,j} &\leftarrow w_{i,j} - \eta\frac{\partial \epsilon}{\partial w_{i,j}}.
\end{aligned}$$

2. a) The second AdaBoost iteration yields an optimal weak classifier using feature 1, threshold $\tau_2 = 0.5$, and polarity -1. Samples 3 and 4 are misclassified, giving an error

$$\varepsilon_2 = \frac{2}{10} = \frac{1}{5} \qquad \alpha_2 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_2}{\varepsilon_2}\right) = \ln(2) \approx 0.69$$

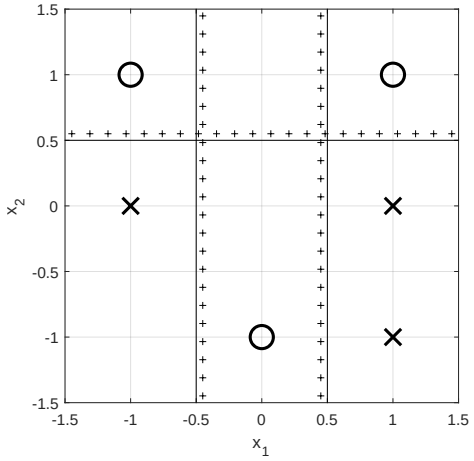
and updated normalized weights

$$\mathbf{d}_2 = \begin{bmatrix} \frac{1}{16} & \frac{5}{16} & \frac{1}{4} & \frac{1}{4} & \frac{1}{16} & \frac{1}{16} \end{bmatrix}$$

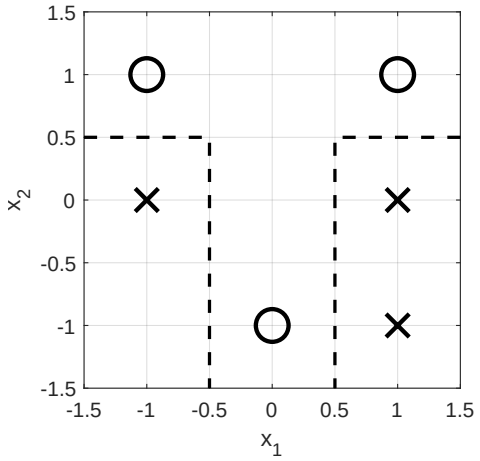
The third iteration yields an optimal weak classifier using feature 1, threshold $\tau_3 = -0.5$, and polarity 1. Samples 1, 5, and 6 are misclassified, giving an error

$$\varepsilon_3 = \frac{3}{16} \qquad \alpha_3 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_3}{\varepsilon_3}\right) = \ln\left(\sqrt{\frac{13}{3}}\right) \approx 0.73$$

b) The three weak classifiers divide the input space into six regions. Comparing the alpha-strength and sign of the three classifiers in the different regions gives the following decision boundary, where all samples are correctly classified:



(a) Weak classifier decision boundaries, with positive sides indicated by the plus markers.



(b) Strong classifier decision boundary.

3. a) First we calculate the covariance matrix

$$C = \frac{1}{N-1}(\mathbf{X}-\mathbf{m})(\mathbf{X}-\mathbf{m})^T = \left[\mathbf{m} = \begin{pmatrix}2\\0\end{pmatrix}\right] = \frac{1}{7}\begin{pmatrix}28 & 14\\14 & 28\end{pmatrix} = \begin{pmatrix}4 & 2\\2 & 4\end{pmatrix}$$

The two eigenvalues are $\lambda_1 = 6$ and $\lambda_2 = 2$, with corresponding normalized eigenvectors

$$\mathbf{e}_1 = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \qquad \mathbf{e}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\end{pmatrix}$$

Project $\mathbf{X}_c = \mathbf{X} - \mathbf{m}$ on $\mathbf{e}_1$:

$$\begin{aligned}\mathbf{Y} = \mathbf{e}_1^T\mathbf{X}_c &= \frac{1}{\sqrt{2}}\begin{pmatrix}-5 & -4 & -2 & 2 & 3 & -1 & 3 & 4\end{pmatrix}\\ &\approx \begin{pmatrix}-3.54 & -2.83 & -1.41 & 1.41 & 2.12 & -0.71 & 2.12 & 2.83\end{pmatrix}\end{aligned}$$

b) The amount of information that is accounted for by the first principle direction is given by

$$\frac{\lambda_1}{\lambda_1+\lambda_2} = \frac{6}{6+2} \approx 75\%,$$

c) $\mathbf{Y}$ are the centered coordinates of the original signal for the base vector $\mathbf{e}_1$. A reconstructed signal $\mathbf{X}_r$ is therefore obtained as

$$\mathbf{X}_r = \mathbf{e}_1\mathbf{Y} + \mathbf{m} = \frac{1}{2}\begin{pmatrix}-1 & 0 & 2 & 6 & 7 & 3 & 7 & 8\\-5 & -4 & -2 & 2 & 3 & -1 & 3 & 4\end{pmatrix}$$

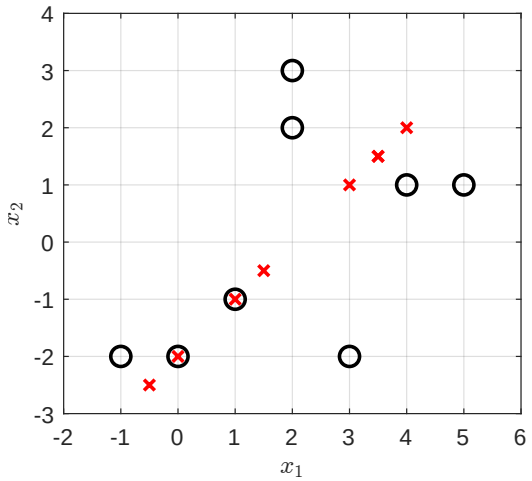


Figure 3: Original and reconstructed data

4. The estimated Q-function is defined by:

$$\hat{Q}(s_k, a_j) \leftarrow (1-\alpha)\hat{Q}(s_k, a_j) + \alpha\Big(r + \gamma \max_a \hat{Q}(s_{k+1}, a)\Big)$$

After the first sequence, the only state that is affected is S_9:

$$\hat{Q}(s_9, left) = (1-\alpha)\cdot 0 + \alpha(5 + \gamma \cdot 0) = 5\alpha$$

The updated Q-function will be:

Q(s,a)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Up	0	0	-	0	0	0	0	*end*	-
Left	-	0	-	0	0	-	0	*end*	5α
Right	0	-	0	0	-	0	-	*end*	-

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

$$\hat{Q}(s_4, up) = (1-\alpha)\cdot 0 + \alpha(-1 + \gamma \cdot 0) = -\alpha$$
$$\hat{Q}(s_6, right) = (1-\alpha)\cdot 0 + \alpha(5 + \gamma \cdot 0) = 5\alpha$$

The updated Q-function will be:

Q(s,a)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Up	0	0	-	$-\alpha$	0	5α	0	*end*	-
Left	-	0	-	0	0	-	0	*end*	5α
Right	0	-	0	0	-	0	-	*end*	-

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

$$\hat{Q}(s_7, up) = (1-\alpha)\cdot 0 + \alpha(0 + \gamma \cdot 5\alpha) = 5\alpha^2\gamma$$
$$\hat{Q}(s_9, up) = (1-\alpha)\cdot 5\alpha + \alpha(5 + \gamma \cdot 0) = 5\alpha - 5\alpha^2 + 5\alpha = 10\alpha - 5\alpha^2 = 5(2\alpha - \alpha^2)$$

The updated Q-function will be:

Q(s,a)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Up	0	0	-	$-\alpha$	0	5α	$5\alpha^2\gamma$	*end*	-
Left	-	0	-	0	0	-	0	*end*	$5(2\alpha - \alpha^2)$
Right	0	-	0	0	-	0	-	*end*	-