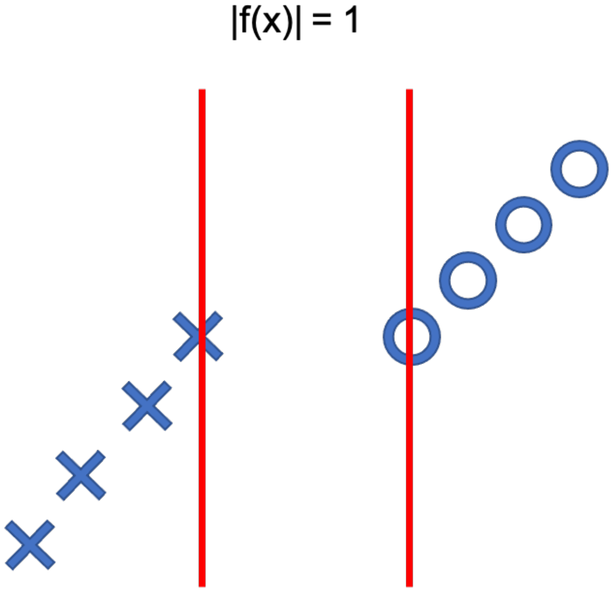
# Solutions to the exam in
# Neural Networks and Learning Systems - TBMI26 / 732A55
# Exam 2021-06-09

Part 1

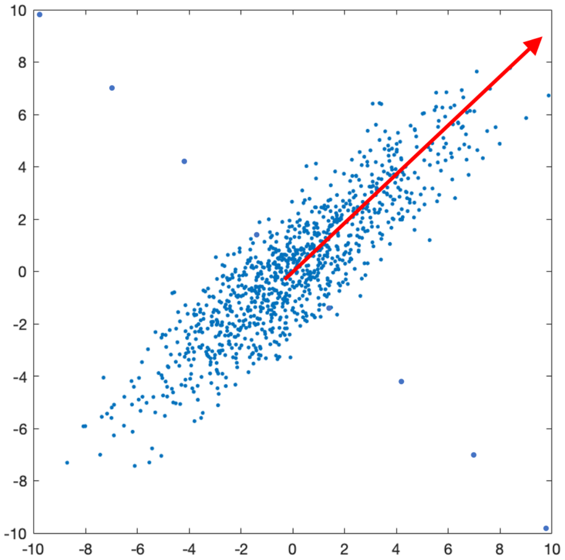
1. $T_1$: Unsupervised learning

   $T_2$: Supervised learning

2. See figure:

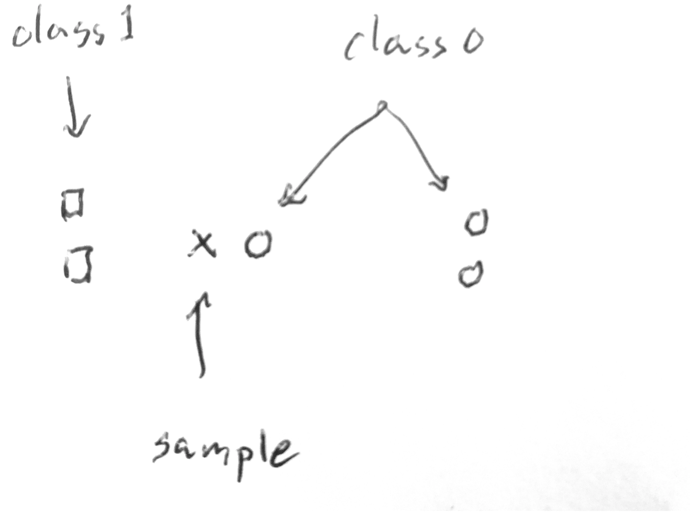


3. Only tanh can be used.
4. See figure:



5. We need to train three parameters. 2 for the two features and one bias weight.
6. The convolution kernels are usually much smaller than the input space (image). This means that the CNN has much fewer parameters to train compared to a fully connected NN, and therefore has a fewer degrees of freedom and, hence, requires fewer training data examples.
7. Image segmentation means that each pixel should be classified as foreground or background.
8. Data augmentation. The purpose is to improve generalization.

9. $\mathbf{K} = \phi^T \phi$.
10. See figure:



11. Confusion matrix:

| | | |
|---|---|---|
| $A$ | 0 | 0 |
| 0 | $B$ | 0 |
| 0 | 3 | $C-3$ |

Accuracy: $\frac{A+B+C-3}{A+B+C}$

12. 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.

13. k-means, where the mean of each cluster is estimated, and Mixture of Gaussian (MoG), where the mean and covariance matrix for each cluster is estimated.

14. The assumption is that the classes have similar distributions.

15. Besides the training data, we use *validation data* for monitoring the generalization error during training and, finally, *test data* for testing the final performance after training.

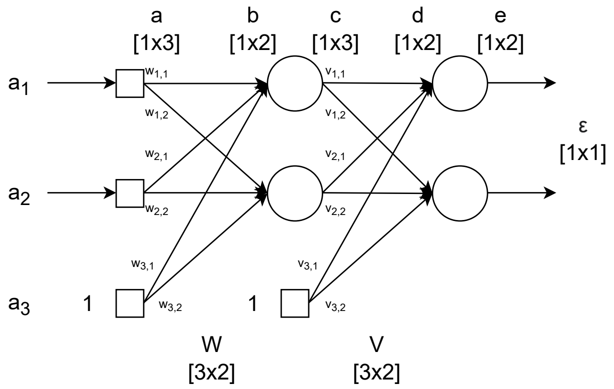


Figure 1: Neural network diagram.

Part 2

1. a) Figure **??** shows a labeled diagram of the network. We define the variables at each stage of the network:

$$a_i = i\text{-th input}, \quad b_j = \sum_{i=1}^{3} a_i w_{i,j},$$

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

$$e_k = d_k \,(\text{no activation}), \quad \epsilon = \sum_{k=1}^{2} (e_k - y_k)^2,$$

where $y_k$ represents the target value for the $k$-th output.

From the above definitions, we calculate all partial derivatives between contiguous stages of the network:

$$\frac{\partial \epsilon}{\partial e_k} = \frac{\partial}{\partial e_k} \sum_{k=1}^{2} (e_k - y_k)^2 = 2(e_k - y_k),$$

$$\frac{\partial e_k}{\partial d_k} = \frac{\partial}{\partial d_k} d_k = 1,$$

$$\frac{\partial d_k}{\partial c_j} = \frac{\partial}{\partial c_j} \sum_{j=1}^{3} c_j v_{j,k} = v_{j,k}, \quad \frac{\partial d_k}{\partial v_{j,k}} = \frac{\partial}{\partial v_{j,k}} \sum_{j=1}^{3} c_j v_{j,k} = c_j,$$

$$\frac{\partial c_j}{\partial b_j} = \frac{\partial}{\partial b_j} b_j = 1, \quad j = 1, 2,$$

$$\frac{\partial b_j}{\partial a_i} = \frac{\partial}{\partial a_i} \sum_{i=1}^{3} a_i w_{i,j} = w_{i,j}, \quad \frac{\partial b_j}{\partial w_{i,j}} = \frac{\partial}{\partial w_{i,j}} \sum_{i=1}^{3} a_i w_{i,j} = a_i.$$

Using these, we can find the gradient of the loss with respect to the weight matrices:

$$\frac{\partial \epsilon}{\partial v_{j,k}} = \frac{\partial \epsilon}{\partial e_k} \frac{\partial e_k}{\partial d_k} \frac{\partial d_k}{\partial v_{j,k}} = 2(e_k - y_k) c_j,$$

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

and the weight update rules are

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

b) Adding tanh activation changes the previous definitions to

$$c_j = \begin{cases} \tanh(b_j) & j = \{1,2\} \\ 1, & j = 3 \end{cases}, \quad e_k = \tanh(d_k).$$

The new partial derivatives are

$$\begin{aligned} \frac{\partial e_k}{\partial d_k} &= \frac{\partial}{\partial d_k} \tanh(d_k) = 1 - \tanh(d_k)^2 = 1 - e_k^2, \\ \frac{\partial c_j}{\partial b_j} &= \frac{\partial}{\partial b_j} \tanh(b_j) = 1 - \tanh(b_j)^2 = 1 - c_j^2, \quad j = 1, 2, \end{aligned}$$

which give the new gradients

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

and the same update rules as before:

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

c) Without activation functions, the first network behaves like a single-layer network, and can only learn linear decision boundaries. The second network can learn non-linear decision boundaries.

2. a) Figure **??**(a) shows the state of the problem after the second AdaBoost iteration and the results of the third AdaBoost iteration. After selecting the threshold yielding minimum error we get $\epsilon_3 = \frac{3}{24} = \frac{1}{8}$ and $\alpha_3 = \frac{1}{2}\ln\frac{1-\epsilon_3}{\epsilon_3} = \frac{1}{2}\ln\frac{7/8}{1/8} = \frac{\ln 7}{2}$. The weights of the correctly classified samples are multiplied by $e^{-\alpha_3} = \frac{1}{\sqrt{7}}$, while the weight of those of the misclassified samples are multiplied by $e^{\alpha_3} = \sqrt{7}$. The sum of the weights is $\frac{7}{8\sqrt{7}} + \frac{\sqrt{7}}{8} = \frac{\sqrt{7}}{4}$, and after normalizing they take the values shown in Figure **??**(a).

   b) We have the following data:

$$\mathbf{C} = \begin{bmatrix}\mathbf{c}_1\\ \mathbf{c}_2\\ \mathbf{c}_3\end{bmatrix} = \begin{bmatrix}1 & 1 & -1 & 1 & 1 & -1 & 1 & -1\\ 1 & -1 & -1 & 1 & -1 & -1 & -1 & -1\\ -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

$$\boldsymbol{\alpha} = \begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix} = \begin{bmatrix}0.97\\ 0.90\\ 0.97\end{bmatrix}$$

The strong classifier results are shown in Figure **??**(c). The strong classifier $\mathbf{H}_i$ is defined as $sign\left(\sum_i \alpha_i \mathbf{c}_i\right)$ where $\mathbf{c}_i$ are the classifications by the weak classifiers.

$$\mathbf{H}_1 = sign(\alpha_1 \mathbf{c}_1) = \begin{bmatrix}1 & 1 & -1 & 1 & 1 & -1 & 1 & -1\end{bmatrix}$$

$$\mathbf{H}_2 = sign(\sum_{i=1}^{2} \alpha_i \mathbf{c}_i) = \begin{bmatrix}1 & 1 & -1 & 1 & 1 & -1 & 1 & -1\end{bmatrix}$$

$$\mathbf{H}_3 = sign(\sum_{i=1}^{3} \alpha_i \mathbf{c}_i) = \begin{bmatrix}1 & -1 & -1 & 1 & 1 & -1 & 1 & -1\end{bmatrix}$$

The strong classifiers from one or two weak classifiers give and accuracy of $\frac{7}{8}$, while the strong classifier from three weak classifiers gives an accuracy of 1.

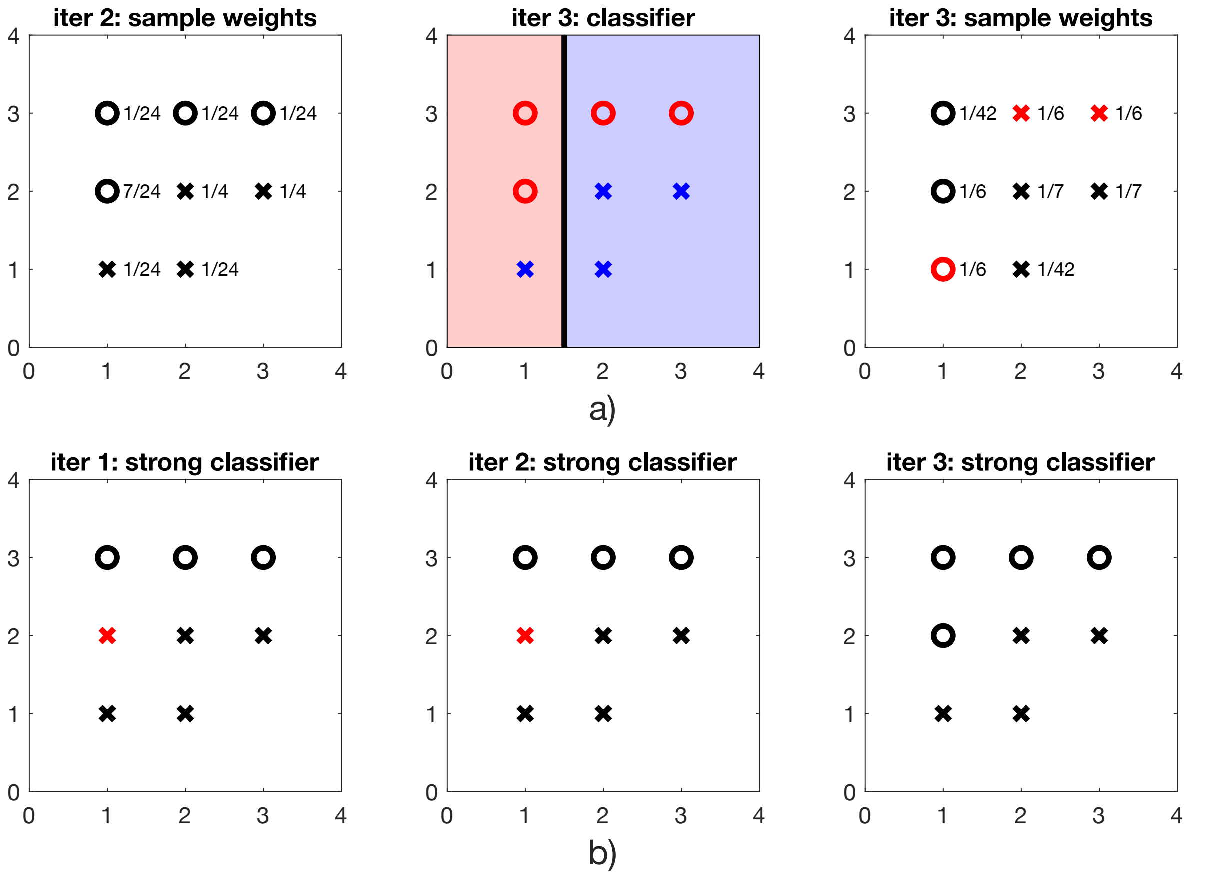


Figure 2: a) Initial state. Blue crosses mark the positive class and red circles mark the negative. b) Left: first threshold. Right: classification and updated weights. Black and red indicate correct and incorrect classifications, respectively. c) Left: second threshold. Middle: classification and updated weights. Black and red indicate correct and incorrect classifications, respectively. Right: strong classifier results. Black and red indicate correct and incorrect classifications, respectively.

3. a) First we estimate the covariance matrix

$$\tilde{\mathbf{C}} = \frac{1}{N-1}\left(\mathbf{x}-\mathbf{m}\right)\left(\mathbf{x}-\mathbf{m}\right)^T = \left[\mathbf{m} = \begin{pmatrix}1\\2\end{pmatrix}\right] = \frac{1}{5-1}\begin{pmatrix}20 & 16\\16 & 20\end{pmatrix} = \begin{pmatrix}5 & 4\\4 & 5\end{pmatrix}.$$

The eigenvalues are $\lambda_1 = 9$ and $\lambda_2 = 1$. The normalized eigenvector to the largest eigenvalue is

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

Project $\bar{\mathbf{x}}(t) = \mathbf{x} - \mathbf{m}$ on $\mathbf{e}_1$ which gives the following dataset:

| t | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $s(t)$ | $-3\sqrt{2}$ | 0 | 0 | 0 | $3\sqrt{2}$ |

b) $s(t)$ are (the centered) coordinates of the original signal for the base vector $\mathbf{e}_1$. A reconstructed signal $\tilde{\mathbf{x}}(t)$ is therefore obtained as $\tilde{\mathbf{x}}(t) = \mathbf{e}_1 s(t) + \mathbf{m}$:

| t | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\tilde{x}_1(t)$ | -2 | 1 | 1 | 1 | 4 |
| $\tilde{x}_2(t)$ | -1 | 2 | 2 | 2 | 5 |

c) We can observe in b) that sample 2-4 are reconstructed to the same point, even though they were separate in the original data. Thus important information is lost, and PCA should most likely not be used as a compression method for this dataset. (However, with a good motivation the opposite answer will also be accepted.)

4. The formula for the optimal Q-function is given by:

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

a)

$$\begin{aligned} V^*(5) &= 0 && (1) \\ V^*(4) &= 0 + \gamma V^*(5) = 0 && (2) \\ Q(2 \to 4) &= 2 + \gamma V^*(4) = 2 && (3) \\ Q(2 \to 5) &= 3 \cdot p + 1 \cdot (1-p) + \gamma V^*(5) = 2p+1 && (4) \\ V^*(2) &= \begin{cases} 2 & \text{if } p < 1/2 \\ 2p+1 & \text{if } p \geq 1/2 \end{cases} && (5) \\ V^*(3) &= 0 + \gamma V^*(4) = 0 && (6) \\ Q(1 \to 3) &= 0 + \gamma V^*(3) = 0 && (7) \\ Q(1 \to 2) &= 1 + \gamma V^*(2) \\ &= \begin{cases} 1+2\gamma & \text{if } p < 1/2 \\ 1+\gamma(2p+1) & \text{if } p \geq 1/2 \end{cases} && (8) \\ V^*(1) &= max(Q(1 \to 3), Q(1 \to 2)) = \\ &= / \text{ given } \gamma \in [0,1] \text{ and } p \in [0,1] \ / = \\ &= \begin{cases} 1+2\gamma & \text{if } p < 1/2 \\ 1+\gamma(2p+1) & \text{if } p \geq 1/2 \end{cases} && (9) \end{aligned}$$

b)

$$Q(s_k,a_i) = (1-\alpha)Q(s_k,a_i) + \alpha(r(s_s,a_i) + \gamma V^*(s_{k+1})) \qquad (10)$$

With $\gamma = 1$ and $p = 0.1$:

$$\begin{aligned} Q(2 \to 5) &= 0 + \alpha \cdot (3p + (1-p) + V^*(5)) = 1.2\alpha && (11) \\ Q(1 \to 2) &= 0 + \alpha \cdot (1 + V^*(2)|_{p=0.1}) = \alpha + 1.2\alpha && (12) \\ Q(2 \to 4) &= 0 + \alpha \cdot (2 + V^*(4)) = 2\alpha && (13) \\ Q(3 \to 4) &= 0 + \alpha \cdot (0 + V^*(4)) = 0 && (14) \\ Q(2 \to 4) &= 2\alpha(1-\alpha) + \alpha \cdot (2 + V^*(4)) = 4\alpha - 2\alpha^2 && (15) \end{aligned}$$