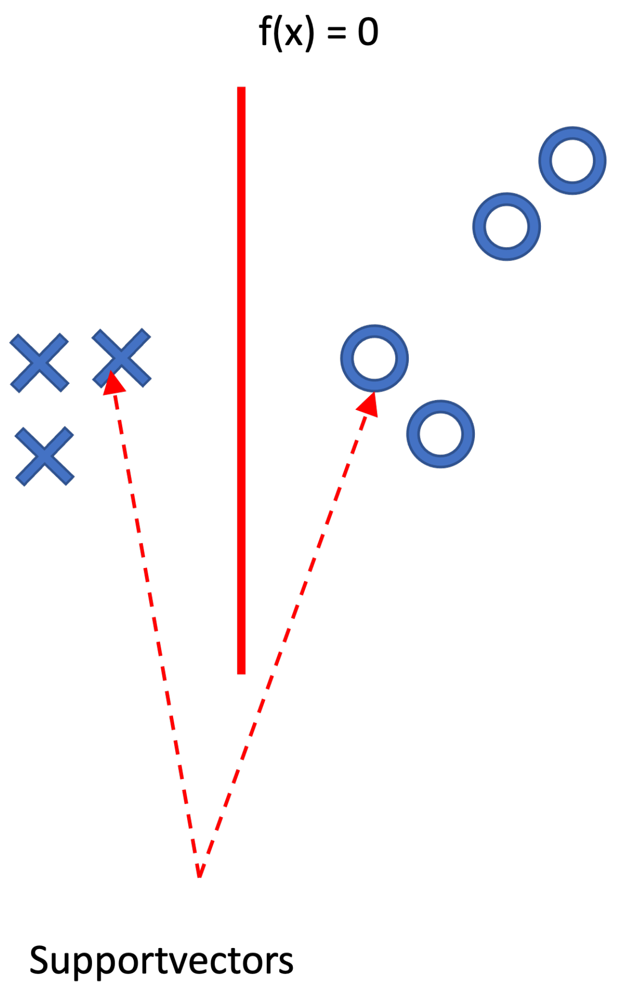
Solutions to the exam in Neural Networks and Learning Systems - TBMI26 / 732A55 Exam 2023-08-26

Part 1

1. - Q-learning - Reinforcement learning
   - Support vector machines - Supervised learning
   - Principal component analysis - Unsupervised learning
2. See figure below



3. A large weight means that the sample has been misclassified frequently by the weak classifiers. Thus, this is a difficult sample to classify, and it might be an outlier that has caused the classifier to overtrain.

4. We need to train three parameters. 2 for the two features and one bias weight.
5. The purpose of the hidden layers in a multi-layer perceptron classifier is transform the data to a space where the problem i linearly separable.
6. The value of k should increase.
7. By data augmentation, i.e. rotating, scaling or shifting the images.
8. The value of the V function for is the maximum Q-value over all possible actions.
9. The general definition of a kernel function is $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$, where $\phi((\mathbf{x})$ is a non-linear function.
10. The accuracy is the number of correctly classified examples in the diagonal of the matrix divided with the total number of samples, i.e., (100+50+150)/400 = 0.75.

Two-point questions

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

12. The ReLU function is linear for positive values and zero for negative values. The advantage is that the gradient is constant and does not decrease when propagated back thru the network. This means that it avoids the so-called "vanishing gradient problem". See lecture for drawings.

13. The convolutional layers in a CNN used fewer parameters that a fully connected network. The convolution als gives shift invariance in feature detection.

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.

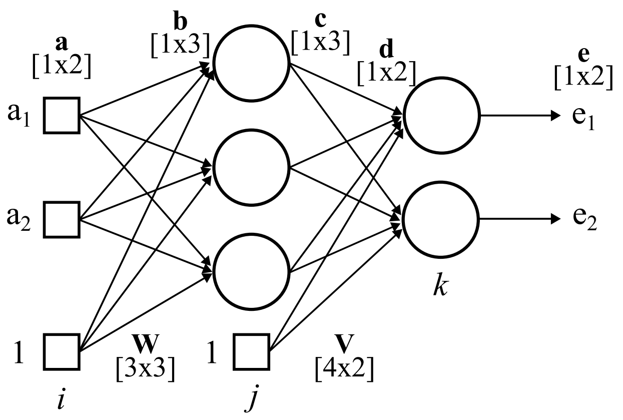


Figure 1: Neural network diagram.

Part 2

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

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

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

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

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:

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

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

$$\frac{\partial d_k}{\partial c_j} = \frac{\partial}{\partial c_j} \sum_{j=1}^{4} c_j v_{j,k} = v_{j,k}, \quad \frac{\partial d_k}{\partial v_{j,k}} = \frac{\partial}{\partial v_{j,k}} \sum_{j=1}^{4} c_j v_{j,k} = c_j,$$

$$\frac{\partial c_j}{\partial b_j} = \frac{\partial}{\partial b_j} b_j = 1, \quad j = 1,2,3,$$

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

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

$$c_j = \begin{cases} \tanh(b_j) & j = \{1,2,3\} \\ 1, & j = 4 \end{cases}, \quad e_k = \tanh(d_k).$$

The new partial derivatives are

$$\begin{aligned} \frac{\partial e_k}{\partial d_k} &= \frac{\partial}{\partial d_k} \tanh(d_k) = 1 - \tanh(d_k)^2 = 1 - e_k^2, \\ \frac{\partial c_j}{\partial b_j} &= \frac{\partial}{\partial b_j} \tanh(b_j) = 1 - \tanh(b_j)^2 = 1 - c_j^2, \quad j = 1,2,3, \end{aligned}$$

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) There are three different week classifiers that each give an error of 3/8, which is optimal in the first iteration. These are

   - $x_1 < -0.5$
   - $x_1 > 0.5$
   - $x_2 > 0.5$

   b) From the figure we define

$$\mathbf{X} = \begin{bmatrix} -1 & -1 & 0 & 0 & -1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & -1 & -1 & -1 & 1 \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

The first AdaBoost iteration yields an optimal weak classifier using feature 2, threshold $\tau_1 = -0.5$, and polarity 1. Sample 8 is misclassified, giving an error

$$\varepsilon_1 = \frac{1}{8} \qquad \alpha_1 = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_1}{\varepsilon_1} \right) = \ln \left(\sqrt{7} \right) \approx 0.97$$

and updated normalized weights

$$\mathbf{d}_1 = \begin{bmatrix} \frac{1}{14} & \frac{1}{14} & \frac{1}{14} & \frac{1}{14} & \frac{1}{14} & \frac{1}{14} & \frac{1}{14} & \frac{1}{2} \end{bmatrix}$$

The second iteration yields an optimal weak classifier using feature 1, threshold $\tau_2 = 0.5$, and polarity -1. Samples 5 and 6 are misclassified, giving an error

$$\varepsilon_2 = \frac{2}{14} = \frac{1}{7} \qquad \alpha_2 = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_2}{\varepsilon_2} \right) = \ln \left(\sqrt{6} \right) \approx 0.90$$

and updated normalized weights

$$\mathbf{d}_2 = \begin{bmatrix} \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{24} & \frac{1}{4} & \frac{1}{4} & \frac{1}{24} & \frac{7}{24} \end{bmatrix}$$

3. a) First we calculate the covariance matrix

$$C = \frac{1}{N-1}(\mathbf{X}-\mathbf{m})(\mathbf{X}-\mathbf{m})^T = \left[\mathbf{m} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right] = \frac{1}{5}\begin{pmatrix} 18 & 4 \\ 4 & 12 \end{pmatrix}$$

The two eigenvalues are $\lambda_1 = 4$ and $\lambda_2 = 2$, with corresponding normalized eigenvectors

$$\mathbf{e}_1 = \frac{1}{\sqrt{5}}\begin{pmatrix} 2 \\ 1 \end{pmatrix} \qquad \mathbf{e}_2 = \frac{1}{\sqrt{5}}\begin{pmatrix} 1 \\ -2 \end{pmatrix}$$

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

$$\begin{aligned} \mathbf{Y} = \mathbf{e}_1^T\mathbf{X}_c &= \frac{1}{\sqrt{5}}\begin{pmatrix} -5 & -3 & -4 & 3 & 4 & 5 \end{pmatrix} \\ &\approx \begin{pmatrix} -2.24 & -1.34 & -1.79 & 1.34 & 1.79 & 2.24 \end{pmatrix} \end{aligned}$$

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

$$\frac{\lambda_1}{\lambda_1 + \lambda_2} = \frac{4}{4+2} \approx 67\%,$$

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

$$\mathbf{X}_r = \mathbf{e}_1\mathbf{Y} + \mathbf{m} = \frac{1}{5}\begin{pmatrix} -10 & -6 & -8 & 6 & 8 & 10 \\ 0 & 2 & 1 & 8 & 9 & 10 \end{pmatrix}$$

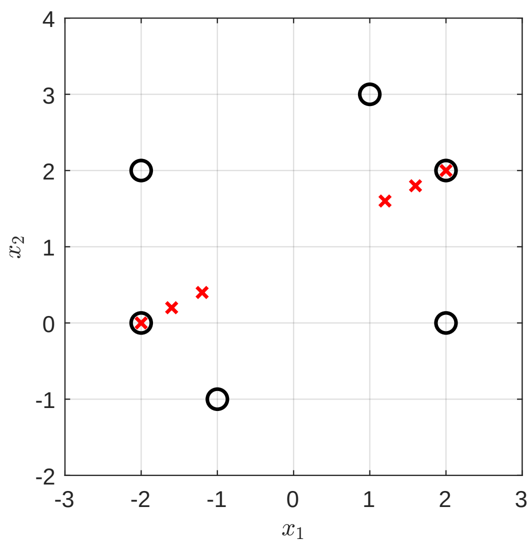


Figure 2: Original and reconstructed data

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

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

and for the V-function:

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

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

a)

$$\begin{aligned}
V^*(3) &= 1 + \gamma V^*(2) \\
V^*(2) &= 0 + \gamma V^*(3) \\
V^*(3) = Q^*(3,2) &= \frac{1}{1-\gamma^2} \\
V^*(2) = Q^*(2,3) &= \frac{\gamma}{1-\gamma^2} \\
V^*(1) = Q^*(1,2) &= \frac{\gamma^2}{1-\gamma^2}
\end{aligned}$$

b)

According to the state model we have :

$$\begin{aligned}
V^*(3) &= 0 \\
V^*(4) &= 0 \\
V^*(5) = Q^*(5,4) &= -8 + \gamma V^*(4) = -8 \\
V^*(1) = Q^*(1,2) &= 0 + \gamma V^*(2)
\end{aligned}$$

For state 2 there are two options:

$$\begin{aligned}
Q^*(2,3) &= 2 + \gamma V^*(3) = 2 \\
Q^*(2,4) &= 3p + (1-p)(3 + \gamma V^*(5)) = 3 - 8\gamma(1-p)
\end{aligned}$$

with the value for $Q^*(2,4)$ given as the weighted sum, by the different probabilities, of the two ways of reaching 4.

The best policy depends on γ and p. $2 \rightarrow 3$ is the optimal policy when:

$$\begin{aligned}
Q^*(2,3) &> Q^*(2,4) \\
2 &> 3 - 8\gamma(1-p) \\
\gamma &> \frac{1}{8(1-p)}
\end{aligned}$$

With this we get:

$$\begin{aligned} V^*(2) &= \begin{cases} 2 & \text{if } \gamma > \frac{1}{8(1-p)} \\ 3 - 8\gamma(1-p) & \text{if } \gamma \leq \frac{1}{8(1-p)} \end{cases} \\ V^*(1) &= \begin{cases} 2\gamma & \text{if } \gamma > \frac{1}{8(1-p)} \\ 3\gamma - 8\gamma^2(1-p) & \text{if } \gamma \leq \frac{1}{8(1-p)} \end{cases} \end{aligned}$$

c) Given any γ, the action $2 \to 3$ will be better than $2 \to 4$ when:

$$p \;<\; 1 - \frac{1}{8\gamma}$$

Thus with $\gamma = 0.5$ we get $p < 0.75$.