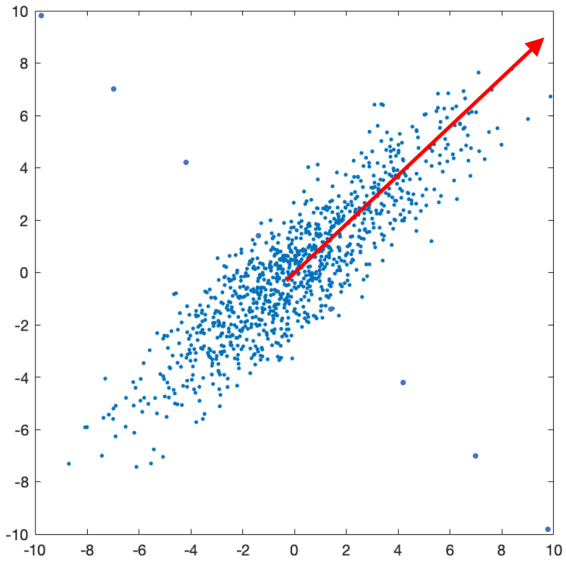
Solutions to the exam in Neural Networks and Learning Systems - TBMI26 / 732A55 Exam 2022-06-09

Part 1

1.
   - Q-learning - Reinforcement learning
   - Support vector machines - Supervised learning
   - Principal component analysis - Unsupervised learning
2. The slack variables are required when there is no line that can separate the classes, i.e., one cannot find a margin if the slack variables are not introduced.
3. The falsely labeled training example can become an outlier which will gain too much weight in the AdaBoost training because it will be misclassified often by the weak classifiers. Thus, this training example may ruin the classifier.
4. See figure:



5. U-net.
6. k is the number of closest stored examples that are used for voting on class membership.
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. The algorithm has overtrained if it performs much better on the training data than on a test data set which it has not seen before.

9. $\mathbf{K} = \phi^T \phi$.

10. The Q-function describes the value for each action in each state. The V-function only describes the value of a state, assuming that the optimal action is taken in each state.

11. 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".

12. (a) can be used for image segmentation as it has the same size of the output layer as the input layer. (b) can be used for image classification as the output is of lower dimensionality than the input

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

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

14. For example this:

-1	-1	-1
2	2	2
-1	-1	-1

15. Divide the data set into equal parts P1, P2 and P3 with 300 samples each. Then train and evaluate 3 times as follows:
    Train using P1 and P2 and evaluate using P3.
    Train using P1 and P3 and evaluate using P2.
    Train using P2 and P3 and evaluate using P1.
    The total performance is the average classification accuracy of the 3 runs.

Part 2

1. a) The figures should include the input, the activation functions for the hidden layers and how the bias weights are incorporated. One point will be subtracted if anything is missing.

We start to defining the input row vector $\mathbf{x}$ as the feature vector with an added bias element. The desired output is stored in the row vector $\mathbf{d}$ (shape $[1\times 2]$). The weights of the hidden layers are stored in the matrix W with shape $[3\times 2]$. The output weights are stored in the matrix V of the same size as W. The output from the hidden layer is denoted $y = \sigma(\mathbf{s}) = tanh(\mathbf{s}) = tanh(\mathbf{x}W)$, to this vector an additional bias element is added before feeding it forward (resulting shape $[1 \times 3]$). The output is denoted $\mathbf{u} = \mathbf{y}V$.

For the output layer the desired update rule is

$$V_{jk}^{n+1} = V_{jk}^{n} - \alpha \frac{\partial \epsilon}{\partial V_{jk}} \tag{1}$$

where ϵ is the square error $\epsilon = \sum_{k=1}^{2} (d_k - u_k)^2$. Using the chain rule we get

$$\frac{\partial \epsilon}{\partial V_{jk}} = \frac{\partial \epsilon}{\partial u_k}\frac{\partial u_k}{\partial V_{jk}} = -2(d_k - u_k)y_j \tag{2}$$

The desired update rules for the hidden layer is

$$W_{ij}^{n+1} = W_{ij}^{n} - \alpha \frac{\partial \epsilon}{\partial W_{ij}} \tag{3}$$

and using the chain rule we get

$$\frac{\partial \epsilon}{\partial W_{ij}} = \sum_{k=1}^{2} \frac{\partial \epsilon}{\partial u_k}\frac{\partial u_k}{\partial y_j}\frac{\partial y_j}{\partial s_j}\frac{\partial s_j}{\partial W_{ij}} \tag{4}$$

$$= \sum_{k=1}^{2} -2(d_k - u_k)V_{jk}(1 - tanh^2(s_j))x_i \tag{5}$$

The matrix forms of the above update rules are:

$$V^{n+1} = V^{n} - 2\alpha \mathbf{y}^{\top}(\mathbf{d} - \mathbf{u}) \tag{6}$$

$$W^{n+1} = W^{n} - 2\alpha \mathbf{a}^{\top}(\mathbf{d} - \mathbf{u})V^{\top} \circ (1 - \mathbf{s}^{\circ 2}) \tag{7}$$

b) Without the activation function the two layers could be combined to a single layer. In practice we would have a single layer network and therefore only capable of linear decision boundaries.

c) The addition of the weight bias allows the model to learn decision boundaries that are not forced to pass through the origin of the feature space learned by the model.

2. a) We have the following data:

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

The initial weights are

$$\mathbf{d}_0 = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}$$

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

$$\varepsilon_1 = \frac{1}{6}$$

$$\alpha_1 = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_1}{\varepsilon_1} \right) = \ln \left(\sqrt{5} \right)$$

and updated normalized weights

$$\mathbf{d}_1 = \begin{bmatrix} \frac{1}{10} & \frac{1}{2} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} \end{bmatrix}$$

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

$$\varepsilon_2 = \frac{2}{10} = \frac{1}{5}$$

$$\alpha_2 = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_2}{\varepsilon_2} \right) = \ln (2)$$

and updated normalized weights

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

b) This is the classic XOR problem, which cannot be solved using 'decision stumps'. The first iteration can only yield a minimum error 0.5, which means the weights do not update. The next iteration will therefore result in the same optimal solution, and the training is therefore stuck.

3. a) First we calculate the covariance matrix

$$C = \frac{1}{N-1}(\mathbf{X}-\mathbf{m})(\mathbf{X}-\mathbf{m})^T = \left[\mathbf{m} = \begin{pmatrix}0\\1\end{pmatrix}\right] = \frac{1}{5}\begin{pmatrix}24 & 3\\ 3 & 16\end{pmatrix}$$

The two eigenvalues are $\lambda_1 = 5$ and $\lambda_2 = 3$, with corresponding normalized eigenvectors

$$\mathbf{e}_1 = \frac{1}{\sqrt{10}}\begin{pmatrix}3\\1\end{pmatrix} \approx \begin{pmatrix}0.95\\0.32\end{pmatrix}$$

$$\mathbf{e}_2 = \frac{1}{\sqrt{10}}\begin{pmatrix}-1\\3\end{pmatrix} \approx \begin{pmatrix}-0.32\\0.95\end{pmatrix}$$

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

$$\begin{aligned}\mathbf{Y} = \mathbf{e}_1^T\mathbf{X}_c &= \frac{1}{\sqrt{10}}\begin{pmatrix}-7 & -9 & 0 & 2 & 4 & 10\end{pmatrix}\\ &\approx \begin{pmatrix}-2.21 & -2.85 & 0 & 0.63 & 1.26 & 3.16\end{pmatrix}\end{aligned}$$

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

$$\frac{\lambda_1}{\lambda_1+\lambda_2} = \frac{5}{5+3} \approx 62.5\%,$$

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

$$\mathbf{X}_r = \mathbf{e}_1\mathbf{Y} + \mathbf{m} = \frac{1}{10}\begin{pmatrix}-21 & -27 & 0 & 6 & 12 & 30\\ 3 & 1 & 10 & 12 & 14 & 20\end{pmatrix}$$

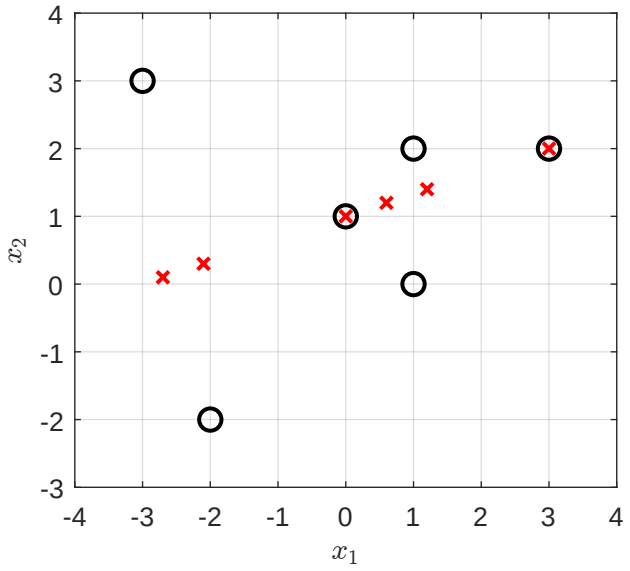


Figure 1: Original and reconstructed data

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

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

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:

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

Combining these two equations yields the following:

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

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:

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

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

$$\begin{aligned} Q^*(2,right) &< Q^*(2,down) \implies && (21)\\ 2\gamma &< 3 + \gamma && (22)\\ & && (23) \end{aligned}$$

This is always true for $0 < \gamma < 1$. Thus:

$$V^*(2) = Q^*(2, down) = 3 + \gamma \tag{24}$$
$$\tag{25}$$

For state 1:

$$Q^*(1, down) = 2 + \gamma V^*(2) = 2 + 3\gamma + \gamma^2 \tag{26}$$
$$Q^*(1, left) = 2 + \gamma V^*(3) = 2 + \gamma \tag{27}$$
$$\tag{28}$$

As before, action "down" in state 1 will be taken if:

$$Q^*(1, right) < Q^*(1, down) \implies \tag{29}$$
$$2 + \gamma < 2 + 3\gamma + \gamma^2 \implies \tag{30}$$
$$\gamma^2 + 2\gamma > 0 \tag{31}$$

This is always true for $0 < \gamma < 1$. Thus:

$$V^*(1) = Q^*(1, down) = 2 + 3\gamma + \gamma^2 \tag{32}$$