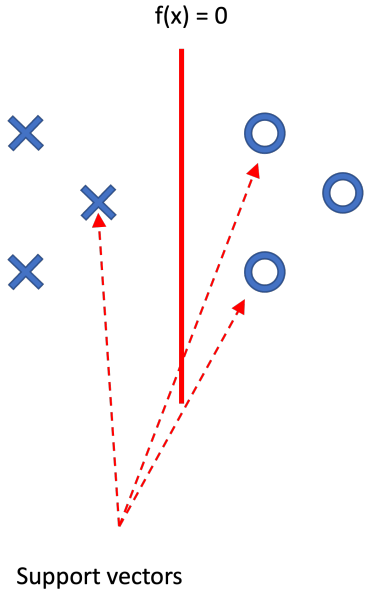
Solutions to the exam in
Neural Networks and Learning Systems - TBMI26 / 732A55
Exam 2023-06-08

Part 1

1. For example:
   Supervised learning - back propagation
   Unsupervised learning - k means
   Reinforcement learning - Q-learning

2. See figure below



3. The problem with outliers cause the weights to grow exponentially, so this problem can be detected by observing the growth of the weights during training.

4. 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.

5. Overfitting can be avoided by reducing the number of parameters in the network or by increasing the amount of training data. It can also be avoided by monitoring the error on validation data and stop training when the validation error is at a minimum. (One answer is enough.)

6. k-NN

7. By data augmentation.

8. The value of the V function for is the maximum Q-value over all possible actions.

9. $$\varphi(\mathbf{x}) = \begin{pmatrix} e^{x_1} \\ e^{x_2} \\ e^{x_3} \end{pmatrix}$$

10. The accuracy is calculated as the number of correctly classified examples divided with the total number of examples. Assume there are x samples in the the first class, then there are $2x$samples in class 2. The accuracy can then be computed as

$$\frac{0.8x + 0.6 \times 2x}{0.8x + 0.6 \times 2x + 0.2x + 0.4 \times 2x} = \frac{0.8 + 1.2}{0.8 + 1.2 + 0.2 + 0.8} = 2/3$$

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. When trining a decision stump in AdaBoost, you use brute force optimization. The reason is that the loss function is piecewise flat and can not be differentiated.

15. In paired training, the trining data consist of paired examples, e.g. a picture and a drawing of the same object. In unpaired trining, you only have unpaired examples from the two domains, e..g. a set of pictures and a set of drawings but of different objects. The advantage with paired training is that it is faster than unpaired training. The advantage with unpaired training is that it is much easier (cheeper) to obtain large amounts of training data, while this can be costly in paired training.

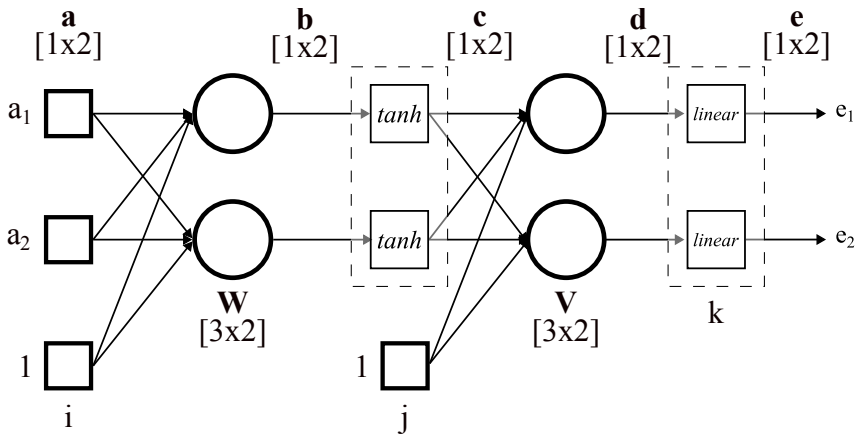


Figure 1: Neural network diagram.

Part 2

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

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

$$c_j = \begin{cases} \tanh(b_j) & j = \{1,2\}, \\ 1, & j = 3 \end{cases} \qquad d_k = \sum_{j=1}^{3} c_j v_{j,k},$$

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

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

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

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

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

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

$$\frac{\partial c_j}{\partial b_j} = \frac{\partial}{\partial b_j} \tanh(b_j) = 1 - \tanh^2(b_j) = 1 - c_j^2, \quad j = 1, 2$$

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

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

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

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

and the weight update rules are

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

c) and d) We are asked to find update rules for the input that maximize specific outputs. We can directly reuse the calculations from b) to find this. To maximize the output of the hidden layer we have

$$\frac{\partial c_j}{\partial a_i} = \frac{\partial c_j}{\partial b_j}\frac{\partial b_j}{\partial a_i} = (1 - \tanh^2(b_j))w_{i,j},$$
$$a_i \leftarrow a_i + \eta \frac{\partial c_j}{\partial a_i},$$

and to maximize the output we have

$$\frac{\partial e_k}{\partial a_i} = \frac{\partial e_k}{\partial d_k}\sum_{j=1}^{2}\frac{\partial d_k}{\partial c_j}\frac{\partial c_j}{\partial b_j}\frac{\partial b_j}{\partial a_i} = \sum_{j=1}^{2} v_{j,k}(1 - \tanh^2(b_j))w_{i,j},$$
$$a_i \leftarrow a_i + \eta \frac{\partial e_k}{\partial a_i}.$$

Note that, since we are maximizing a function, we follow the positive direction of the gradient. Therefore, the gradient is added rather than subtracted from a_i. Note also that we need to sum over j because of the multiple ways in which each e_k depends on each a_i.

2. a) The first AdaBoost iteration yields an optimal weak classifier using feature 2, threshold $\tau_1 = -0.5$, and polarity 1. Sample 6 is misclassified, giving an error

$$\varepsilon_1 = \frac{1}{6} \qquad \alpha_1 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) = \ln\left(\sqrt{5}\right) \approx 0.80$$

and updated normalized weights

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

The second iteration yields an optimal weak classifier using feature 1, threshold $\tau_2 = 1.5$, and polarity 1. Samples 1 and 2 are misclassified, giving an error

$$\varepsilon_2 = \frac{2}{10} = \frac{1}{5} \qquad \alpha_2 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_2}{\varepsilon_2}\right) = \ln\left(\sqrt{4}\right) = \ln(2) \approx 0.69$$

and updated normalized weights

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

b) With only two weak classifiers the one with the lowest error will dominate and fully determine the strong classifier, thus the accuracy is $5/6 \approx 83\%$.

3. The direction that optimally separates the two classes is given by

$$\mathbf{w} = \mathbf{C}_{\text{tot}}^{-1} (\mathbf{m}_{\text{x}} - \mathbf{m}_{\text{o}})$$

where $\mathbf{C}_{\text{tot}}$ is the sum of the individual covariance matrices for the two respective classes, and $\mathbf{m}_{\text{x}}$ and $\mathbf{m}_{\text{o}}$ are the centers of the two classes. We begin with the "crosses" class:

$$\mathbf{X}_{\text{x}} = \begin{pmatrix} -2 & -2 & -1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 & 2 & -2 \end{pmatrix}$$

$$\mathbf{C}_{\text{x}} = \frac{1}{N_{\text{x}} - 1} (\mathbf{X}_{\text{x}} - \mathbf{m}_{\text{x}}) (\mathbf{X}_{\text{x}} - \mathbf{m}_{\text{x}})^T = \left[\mathbf{m}_{\text{x}} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right] = \frac{1}{5} \begin{pmatrix} 4 & -3 \\ -3 & 10 \end{pmatrix}$$

Now for the "circles" class:

$$\mathbf{X}_{\text{o}} = \begin{pmatrix} 1 & 1 & 2 & 0 & 1 & 1 \\ 0 & -2 & 0 & 1 & -1 & 2 \end{pmatrix}$$

$$\mathbf{C}_{\text{o}} = \frac{1}{N_{\text{o}} - 1} (\mathbf{X}_{\text{o}} - \mathbf{m}_{\text{o}}) (\mathbf{X}_{\text{o}} - \mathbf{m}_{\text{o}})^T = \left[\mathbf{m}_{\text{o}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] = \frac{1}{5} \begin{pmatrix} 2 & -1 \\ -1 & 10 \end{pmatrix}$$

We now get:

$$\mathbf{C}_{\text{tot}} = \mathbf{C}_{\text{x}} + \mathbf{C}_{\text{o}} = \frac{2}{5} \begin{pmatrix} 3 & -2 \\ -2 & 10 \end{pmatrix}$$

$$\mathbf{C}_{\text{tot}}^{-1} = \frac{1}{\left(\frac{2}{5}\right)^2 (3 * 10 - 2 * 2)} * \frac{2}{5} \begin{pmatrix} 10 & 2 \\ 2 & 3 \end{pmatrix} = \frac{5}{52} \begin{pmatrix} 10 & 2 \\ 2 & 3 \end{pmatrix}$$

$$\mathbf{w} = \mathbf{C}_{\text{tot}}^{-1} (\mathbf{m}_{\text{x}} - \mathbf{m}_{\text{o}}) = \frac{5}{52} \begin{pmatrix} 20 \\ 4 \end{pmatrix} = \frac{20}{52} \begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

Normalize $\mathbf{w}$:

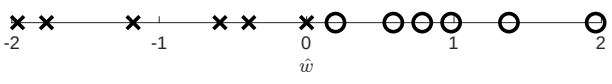
$$\hat{\mathbf{w}} = \frac{\mathbf{w}}{||\mathbf{w}||} = \frac{1}{\sqrt{26}} \begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

Project the data on $\hat{\mathbf{w}}$ which gives the following:

$$\mathbf{Y}_{\text{x}} = \hat{\mathbf{w}}^T \mathbf{X}_{\text{x}} = \frac{1}{\sqrt{26}} \begin{pmatrix} -10 & -9 & -6 & 0 & -3 & -2 \end{pmatrix}$$

$$\mathbf{Y}_{\text{o}} = \hat{\mathbf{w}}^T \mathbf{X}_{\text{o}} = \frac{1}{\sqrt{26}} \begin{pmatrix} 5 & 3 & 10 & 1 & 4 & 7 \end{pmatrix}$$

The projected data is shown in the figure below.



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

$$\begin{aligned} Q(s_k, a) &= r(s_k, a) + \gamma V(s_{k+1}) \\ &= r(s_k, a) + \gamma \max_a Q(s_{k+1}, a) \end{aligned} \tag{1}$$

and for the V-function:

$$V(s_k) = \max_a Q(s_k, a) \tag{2}$$

Note that if only one action is possible for a state, then we have that

$$V(s_k) = Q(s_k, a) \tag{3}$$

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

$$\begin{aligned} Q(1, down) &= V(1) = 1 + \gamma V(2) \\ Q(2, left) &= V(2) = 1 + \gamma V(3) \\ Q(3, right) &= V(4) = 0 + \gamma V(4) \\ Q(4, up) &= V(4) = 2 + \gamma V(2) \end{aligned} \tag{4}$$

Using these equations and starting by substituting for $V(2)$

$$\begin{aligned} V(2) &= 1 + \gamma V(3) \\ &= 1 + \gamma \left(0 + \gamma V(4)\right) \\ &= 1 + \gamma \left(0 + \gamma \left(2 + \gamma V(2)\right)\right) \\ &= 1 + 2\gamma^2 + \gamma^3 V(2) \\ &= \frac{1 + 2\gamma^2}{1 - \gamma^3} \end{aligned} \tag{5}$$

And substitution for all other s_k values:

$$\begin{aligned} V(1) &= 1 + \gamma \frac{1 + 2\gamma^2}{1 - \gamma^3} \\ V(3) &= 2\gamma + \gamma^2 \frac{1 + 2\gamma^2}{1 - \gamma^3} \\ V(4) &= 2 + \gamma \frac{1 + 2\gamma^2}{1 - \gamma^3} \end{aligned} \tag{6}$$

System B:

$$\begin{aligned} V(4) &= 0 \\ Q(3, down) &= V(3) = 4 + \gamma V(4) = 4 \\ Q(2, down) &= 0 + \gamma V(3) = 4\gamma \\ Q(2, right) &= 1 + \gamma V(4) = 1 \end{aligned} \tag{7}$$

To find $V(2)$ we need to find $\max_a(4\gamma, 1)$, from which we get:

$$V(2) = \begin{cases} 4\gamma & \text{for } \gamma \geq 0.25 \\ 1 & \text{for } \gamma < 0.25 \end{cases} \tag{8}$$

Continuing on the remaining s_k:

$$\begin{aligned} Q(1, down) = 2 + \gamma V(2) = \begin{cases} 2 + 4\gamma^2 & \text{for } \gamma \geq 0.25 \\ 2 + \gamma & \text{for } \gamma < 0.25 \end{cases} \\ Q(1, left) = 1 + \gamma V(3) = 1 + 4\gamma \end{aligned} \tag{9}$$

To find $V(1)$ we need to find the $max(Q(1, down), Q(1, left))$, considering the γ value ranges. For $\gamma \geq 0.25$, $Q(1, down) = 2 + 4\gamma^2$, and we have to solve:

$$\begin{aligned} 2 + 4\gamma^2 \geq 1 + 4 \implies \\ 4\gamma^2 - 4\gamma + 1 \geq 0 \end{aligned} \tag{10}$$

Which results (by computing the roots of the quadratic equation) in a double root at $\frac{1}{2}$. This means that $2 + 4\gamma^2$ is always greater than $1 + 4\gamma$ (equal for $\gamma = 0.5$). Thus, for $\gamma \geq 0.25$, $V(1) = 2 + 4\gamma^2$.

For $\gamma < 0.25$ $Q(1, left) = 2 + \gamma$, and we have to solve:

$$\begin{aligned} 2 + \gamma \geq 1 + 4\gamma \implies \\ \gamma \leq 0.33 \end{aligned} \tag{11}$$

Since we are considering the case in which $\gamma < 0.25$, $V(1) = 2 + \gamma$. Finally,

$$V(1) = \begin{cases} 2 + 4\gamma^2 & \text{for } \gamma \geq 0.25 \\ 2 + \gamma & \text{for } \gamma < 0.25 \end{cases} \tag{12}$$