Solutions to the exam in
Neural Networks and Learning Systems - TBMI26 / 732A55
Exam 2020-08-29

Part 1

1. - Q-leaerning - Reinforcement learning
   - k-means - Unsupervised learning
   - kNN (k nearest neighbors) - Supervised learning
2. Support vector machines (SVM).
3. Which of these functions can be used in the hidden layers of a back-prop network?
   - $y = s$ - No, the activation function in a hidden layer needs to be non-linear to be useful
   - $y = \tanh(s)$ - Yes
   - $y = \frac{s}{\|s\|}$ - No, this function is not differentiable
   - $y = e^{(-s^2)}$ - Yes
4. They are (approximately) equal.
5. 3

6. 6

7. The kernel funtion defines a scalar product between the vectors mapped to a (usually high-dimensional) feature space.

$$k(\mathbf{x}_i, \mathbf{x}_j) =< \varphi(\mathbf{x}_i) \mid \varphi(\mathbf{x}_j) >$$

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. k-NN

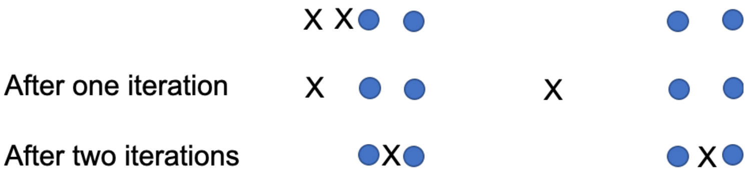
10. 4 times, each time with 75 examples used for training and 25 for evaluation.

11. 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 $\Phi(\mathbf{x})$.

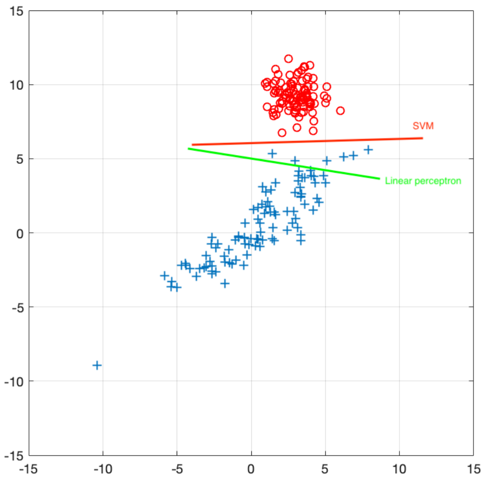
    The distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ in the new feature space is

$$\begin{aligned}
\|\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2)\| &= \sqrt{(\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2))^T \left(\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2)\right)} \\
&= \sqrt{\Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_1) - 2\Phi(\mathbf{x}_1)^T \Phi(\mathbf{x}_2) + \Phi(\mathbf{x}_2)^T \Phi(\mathbf{x}_2)} \\
&= \sqrt{\kappa(\mathbf{x}_1, \mathbf{x}_1) - 2\,\kappa(\mathbf{x}_1, \mathbf{x}_2) + \kappa(\mathbf{x}_2, \mathbf{x}_2)} \\
&= \sqrt{5 - 4 + 2} = \sqrt{3}
\end{aligned}$$

12. The algorithm has converged after the two steps shown below

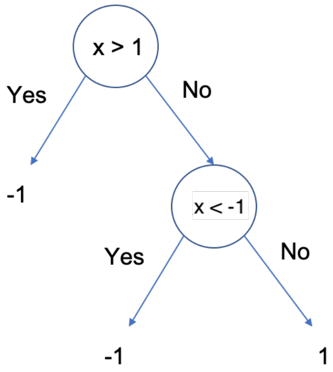


13. We want to train a simple classifier on the data in the figure below. Draw the approximate zero-crossing of the resulting discriminant function if (a) a linear perceptron is used and (b) a linear SVM is used.



14. ReLU = "Rectified Linear Unit", which means that y = max(s,0).
    a) The ReLU does not suffer from the "vanishing gradient" problem in deep networks. It is also computationally efficient. b) The "knockout problem": A neuron can be driven to a state where it never activates for any input.

15.



Part 2

1. a)

C =

6	2	5	2
4	9	2	2
6	4	3	0

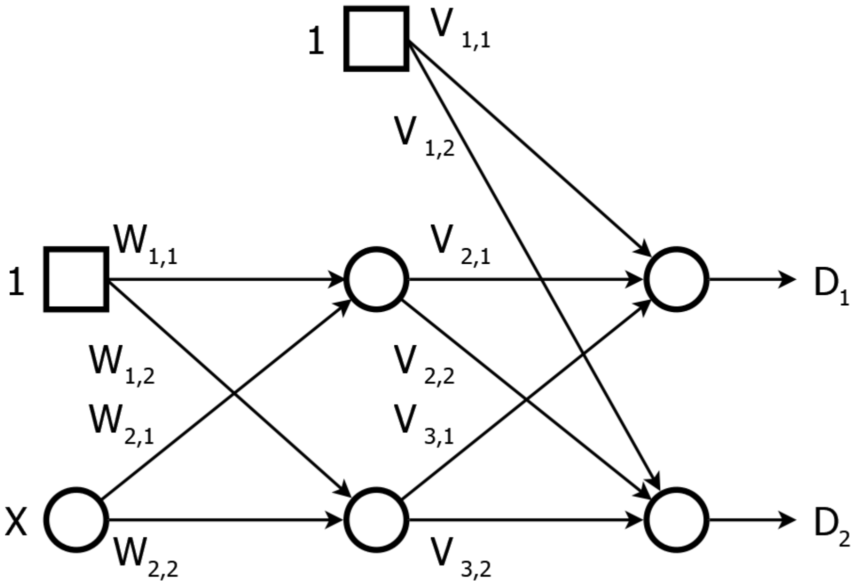
b)

AA =

0	0	0	0	0	0
0	**0**	1	0	1	0
0	2	0	1	0	0
0	0	2	0	0	0
0	0	0	0	0	0

c) Image width (or height) after first correlation: $256 - 3 + 1 = 254$. The activation does not change the size. After pooling: $\lceil 254/2 \rceil = 127 = 2^7 - 1$. After second correlation: $127 - 3 + 1 = 125$. After pooling: $\lceil 125/2 \rceil = 63 = 2^6 - 1$. The image size becomes $1 = 2^1 - 1$ after $N = 7$ complex layers.

2. One possible solution is the network illustrated below. (The number of solutions are, in fact, infinite.)



where

$$\mathbf{W} = \begin{pmatrix} -1 & 0 \\ -1 & 1 \end{pmatrix}, \mathbf{V} = \begin{pmatrix} 0.5 & -0.5 \\ -1 & 1 \\ -1 & 1 \end{pmatrix}$$

After forward propagation through the network with *step* as activation function in the hidden layer and in the output layer, predicted output $\mathbf{D}$ will be:

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}^T$$

exactly corresponding to the classes in $\mathbf{Y}$.

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

$$\mathbf{W} = \mathbf{C}_{tot}^{-1}\left(\mathbf{m}_1 - \mathbf{m}_2\right)$$

where $\mathbf{C}_{tot}$ is the sum of the individual covariance matrices for the two respective classes, and $\mathbf{m}_1$ and $\mathbf{m}_2$ are the centers of the two classes. We begin with the "crosses" class:

$$\mathbf{X}_1 = \begin{pmatrix} -2 & -2 & -2 & -1 & 0 & 1 \\ -1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{C}_1 = \frac{1}{N-1}\left(\mathbf{X}_1 - \mathbf{m}_1\right)\left(\mathbf{X}_1 - \mathbf{m}_1\right)^T = \left[\mathbf{m}_1 = \frac{1}{2}\begin{pmatrix} -2 \\ 1 \end{pmatrix}\right] = \frac{1}{10}\begin{pmatrix} 16 & 6 \\ 6 & 7 \end{pmatrix}$$

Now for the "circles" class:

$$\mathbf{X}_2 = \begin{pmatrix} -1 & 0 & 1 & 2 & 2 & 2 \\ -1 & -1 & -1 & -1 & 0 & 1 \end{pmatrix}$$

$$\mathbf{C}_2 = \frac{1}{N-1}\left(\mathbf{X}_2 - \mathbf{m}_2\right)\left(\mathbf{X}_2 - \mathbf{m}_2\right)^T = \left[\mathbf{m}_1 = \frac{1}{2}\begin{pmatrix} 2 \\ -1 \end{pmatrix}\right] = \frac{1}{10}\begin{pmatrix} 16 & 6 \\ 6 & 7 \end{pmatrix}$$

Note that the covariance matrices are identical due to the symmetry of the two classes. We now get:

$$\mathbf{C}_{tot} = \mathbf{C}_1 + \mathbf{C}_2 = \frac{1}{5}\begin{pmatrix} 16 & 6 \\ 6 & 7 \end{pmatrix}$$

$$\mathbf{C}_{tot}^{-1} = \frac{5*5}{16*7-6*6} * \frac{1}{5}\begin{pmatrix} 7 & -6 \\ -6 & 16 \end{pmatrix} = \frac{5}{76}\begin{pmatrix} 7 & -6 \\ -6 & 16 \end{pmatrix} \approx \begin{pmatrix} 0.46 & -0.39 \\ -0.39 & 1.05 \end{pmatrix}$$

$$\mathbf{W} = \mathbf{C}_{tot}^{-1}\left(\mathbf{m}_1 - \mathbf{m}_2\right) = \frac{5}{19}\begin{pmatrix} -5 \\ 7 \end{pmatrix} \approx \begin{pmatrix} -1.32 \\ 1.84 \end{pmatrix}$$

Normalize $\mathbf{W}$:

$$\hat{\mathbf{W}} = \frac{\mathbf{W}}{||\mathbf{W}||_2} \approx \frac{1}{2.26}\begin{pmatrix} -1.32 \\ 1.84 \end{pmatrix} \approx \begin{pmatrix} -0.58 \\ 0.81 \end{pmatrix}$$

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

$$\mathbf{Y}_1 = \hat{\mathbf{W}}^T\mathbf{X}_1 \approx \begin{pmatrix} 0.35 & 1.16 & 1.98 & 1.40 & 0.81 & 0.23 \end{pmatrix}$$

$$\mathbf{Y}_2 = \hat{\mathbf{W}}^T\mathbf{X}_2 \approx \begin{pmatrix} -0.23 & -0.81 & -1.40 & -1.98 & -1.16 & -0.35 \end{pmatrix}$$

The projected data is shown in the figure below. Again, note the symmetry.

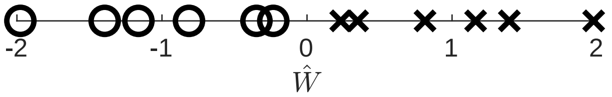


Figure 1: Data projected on $\hat{\mathbf{W}}$

4. The estimated Q-function is defined by: $\hat{Q}(s_k, a_j) \leftarrow (1-\alpha)\hat{Q}(s_k, a_j) + \alpha\Big(r + \gamma \max_a \hat{Q}(s_{k+1}, a)\Big)$

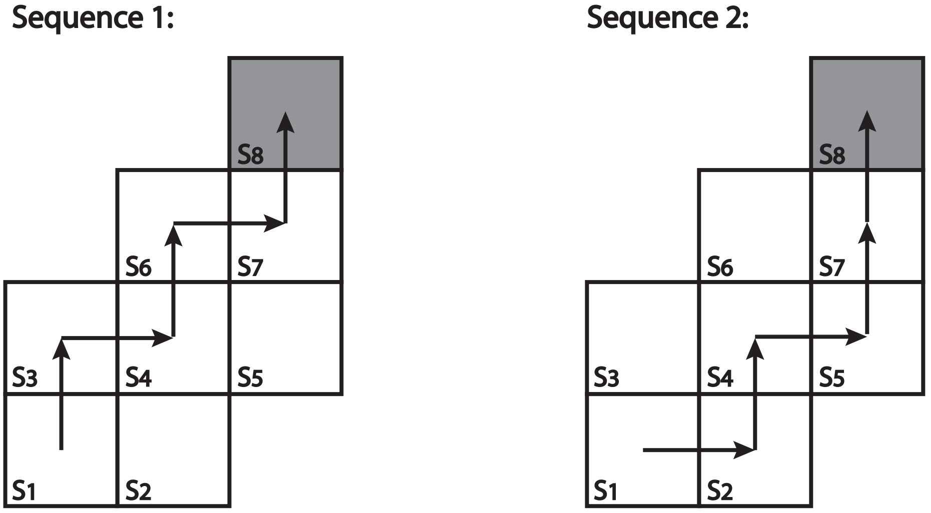


Figure 2: Sequences of action

After the first sequence, the only state that is affected is s_7:

$$\hat{Q}(s_7, up) = (1-\alpha)\cdot 0 + \alpha(15 + \gamma \cdot 0) = 15\alpha$$

The updated Q-function will be:

Q(s,a)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Right	0	-	0	0	-	0	-	*end*
Up	0	0	-	0	0	-	15α	*end*

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

$$\hat{Q}(s_4, right) = (1-\alpha)\cdot 0 + \alpha(-5 + \gamma \cdot 0) = -5\alpha$$
$$\hat{Q}(s_5, up) = (1-\alpha)\cdot 0 + \alpha(0 + \gamma \cdot 15\alpha) = 15\alpha^2\gamma$$
$$\hat{Q}(s_7, up) = (1-\alpha)\cdot 15\alpha + \alpha(15 + \gamma \cdot 0) = 15\alpha(2-\alpha)$$

The updated Q-function will be:

Q(s,a)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Right	0	-	0	-5α	-	0	-	*end*
Up	0	0	-	0	$15\alpha^2\gamma$	-	$15\alpha(2-\alpha)$	*end*

After the third sequence (sequence 1 again), the following states are updated accordingly:

$$\hat{Q}(s_6, right) = (1-\alpha)\cdot 0 + \alpha(0 + \gamma \cdot 15\alpha(2-\alpha)) = 15\alpha^2\gamma(2-\alpha)$$
$$\hat{Q}(s_7, up) = (1-\alpha)\cdot 15\alpha(2-\alpha) + \alpha(15 + \gamma \cdot 0) = 15\alpha(3 - 3\alpha + \alpha^2)$$

The updated Q-function will be:

Q(s,a)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Right	0	-	0	-5α	-	$15\alpha^2\gamma(2-\alpha)$	-	*end*
Up	0	0	-	0	$15\alpha^2\gamma$	-	$15\alpha(3-3\alpha+\alpha^2)$	*end*