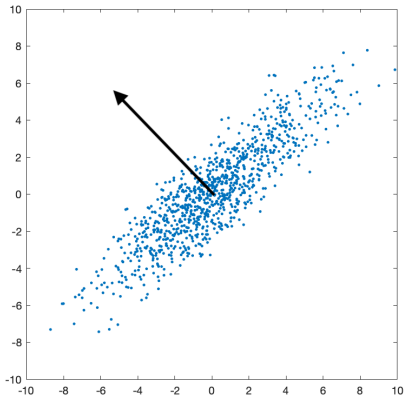
Solutions to the exam in Neural Networks and Learning Systems - TBMI26 / 732A55 Exam 2020-03-20

Part 1

1. The following methods are unsupervised learning methods:

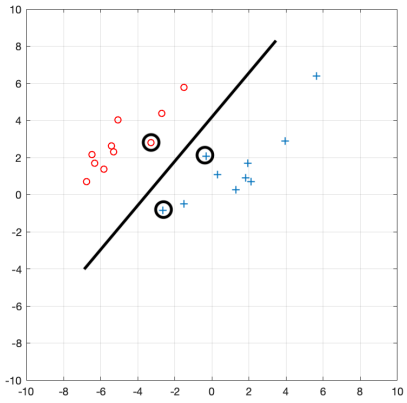
   - Mixture of Gaussians
   - k-means
   - PCA

2. ± 1
3. It must be non-linear and differentiable.
4. The second principal component:



5. There are three parameters to train. Two input weights and one bias weight.

6. A plausible separating line and the support vectors for a linear SVN in the figure below



7. Wee need two mean vectors (each 3 parameters) and two covariance matrices (9 parameters each) which gives $6 + 18 = 24$ parameters in total. (Actually, since the covariance matrices are symmetric we only need to estimate 6 parameters for each covariance matrix, which gives 18 in total.)

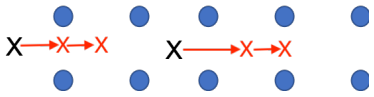
8. The corresponding kernel function $k(x, y) = \sin(x)\sin(y) + \cos(x)\cos(y)$

9. The generalization error is tested on data not used for training.

10. The value of k should increase.

11. A Gaussian or quadratic kernel would be appropriate

12. The algorithm has converged after these two steps:



13. Yes, they are linearly separable. But linear discriminant analysis will perform poorly on this task, since the two classes have very different distributions and LDA assumes similar distributions of the classes.

14. One possible implementation of the decision tree:

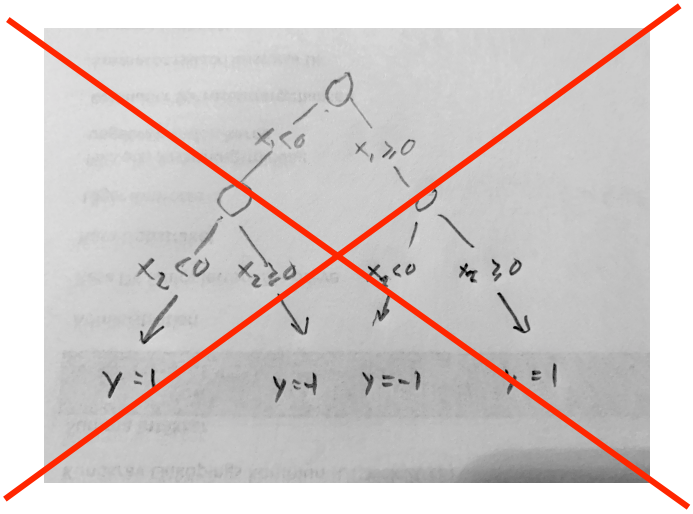


Figure 1: Decision tree

15. The values are $V(S_1) = 3$, $V(S_2) = 5$, $V(S_3) = 7$, and $V(S_4) = 4$. The system will take action A_2.

Part 2

16. a)

g_{00}	=	**1**	2	3	2	(0.5p)
g_{01}	=	**3**	1	3	1	(0.5p)
g_{10}	=	**4**	4	2	2	(0.5p)
g_{11}	=	**4**	3	2	1	(0.5p)

b) i) ☐ $(I + J) \cdot K$ ☐ $I \cdot J + K$ **X** $I \cdot J \cdot K$ ☐ $I \cdot J \cdot (K - 1)$ (1p)
ii) ☐ $J + 2 \cdot K \cdot I$ **X** $J \cdot (2 \cdot K \cdot I - 1)$ ☐ $J \cdot K \cdot I$ ☐ $J \cdot (K \cdot I - 1)$ (1p)

c) 1: c) (0.5p) 2: a) (0.5p) 3: b) (0p)

17. 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.

a)

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

For the output layer the desired update rule is $V_{ki}^{n+1} = V_{ki}^{n} - \alpha\frac{\partial\epsilon}{\partial V_{kj}}$, ϵ is the square error $\epsilon = \sum_{k=1}^{2}(d_k - u_k)^2$. Using the chain rule we get $\frac{\partial\epsilon}{\partial V_{kj}} = \frac{\partial\epsilon}{\partial u_k}\frac{\partial u_k}{\partial V_{ki}} = -2(d_k - u_k)y_i$.

The desired update rules for the hidden layer $W_{ij}^{n+1} = W_{ij}^{n} - \alpha\frac{\partial\epsilon}{\partial W_{ij}}$ and $\frac{\partial\epsilon}{\partial W_{ij}} = \sum_{k=1}^{2}\frac{\partial\epsilon}{\partial u_k}\frac{\partial u_k}{\partial y_i}\frac{\partial y_i}{\partial s_i}\frac{\partial s_i}{\partial W_{ij}} = \sum_{k=1}^{2} -2(d_k - u_k)V_{ki}(1 - tanh^2(s_i))x_j$.

b)

If we put the new hidden layer first in the network we can redefine $\mathbf{x}$ as the output plus bias of the new hidden layer. This will allow us to reuse the update rules from above. Thus all we need is the rule for the new weights.

We store the new weights in the matrix P with two rows and three columns, the features plus bias we call $\mathbf{z}$ and the output from the new hidden layer $\mathbf{x} = tanh(\mathbf{t}) = tanh(P\mathbf{z})$. Now we seek $P_{jm}^{n+1} = P_{jm}^{n} - \alpha\frac{\partial\epsilon}{\partial P_{jm}}$.

$$\frac{\partial\epsilon}{\partial P_{jm}} = \sum_{k=1}^{2}\sum_{i=1}^{3}\frac{\partial\epsilon}{\partial u_k}\frac{\partial u_k}{\partial y_i}\frac{\partial y_i}{\partial s_i}\frac{\partial s_i}{\partial x_j}\frac{\partial x_j}{\partial t_j}\frac{\partial t_j}{\partial P_{jm}} = \sum_{k=1}^{2}\sum_{i=1}^{3} -2(d_k - u_k)V_{ki}(1 - tanh^2(s_i))W_{ij}(1 - tanh^2(t_j))z_m$$

18. a) First we estimate the covariance matrix

$$\tilde{\mathbf{C}} = \mathbb{E}\left[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^T\right] = \quad (1)$$

$$= \frac{1}{N-1}\sum_{t=1}^{5}(\mathbf{x}_t - \mathbf{m})(\mathbf{x}_t - \mathbf{m})^T = \left[\mathbf{m} = \begin{pmatrix}5\\0\end{pmatrix}\right] = \frac{1}{5-1}\begin{pmatrix}26 & -8\\-8 & 6\end{pmatrix} \quad (2)$$

The normalized eigenvector to the largest eigenvalue ($\lambda \approx 7.2$) is

$$\mathbf{e}_1 = \begin{pmatrix}-0.94\\0.33\end{pmatrix}$$

Project $\bar{\mathbf{x}}_t = \mathbf{x}_t - \mathbf{m}$ on $\mathbf{e}_1$: $s(t) = \bar{\mathbf{x}}_t^T\mathbf{e}_1$ which gives the following data set:

t	1	2	3	4	5
$s(t)$	1.89	-2.83	3.49	-2.22	-0.33

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

$$\frac{\lambda_1}{\lambda_1 + \lambda_2} \approx \frac{7.20}{7.20 + 0.80} = 0.9, \quad (3)$$

where λ_1 and λ_2 are the eigenvalues of the first and second principle components, respectively. Thus 90% of the information in the data is accounted for by the first principle direction.

c) $s(t)$ 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 $\tilde{\mathbf{x}}_t = s(t)\mathbf{e}_1 + \mathbf{m}$:

t	1	2	3	4	5
$\tilde{x}_1(t)$	3.2	7.7	1.7	7.1	5.3
$\tilde{x}_2(t)$	0.6	-0.9	1.2	-0.7	-0.1

19. 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:

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

The best policy depends on γ and p. $2 \to 3$ 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:

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

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