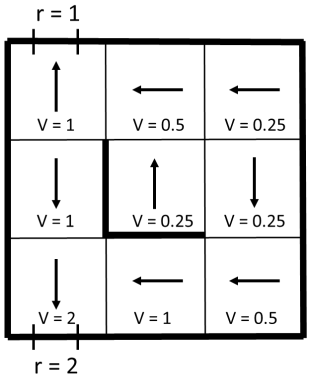
Solutions to the exam in Neural Networks and Learning Systems - TBMI26 Exam 2014-03-20

Part 1

1. Face detection algorithms are typically implemented using an ensemble classifier, i.e., supervised learning.

2. If features are correlated it means that they carry redundant information, i.e., we could perhaps reduce the number of features using dimension reduction. Having more features than necessary means that more noise is introduced in the training, more parameters may be required in the classifier which leads to longer training times, risk of overfitting and potentially more local optima.

3. The value function is shown in the figure below for each state.



4. The weights in the layers are pre-trained using unsupervised learning before the backpropagation algorithm is applied.

5. Problems where the variables are discrete, where we cannot differentiate and use gradient descent methods, are potential candidates for genetic algorithms.

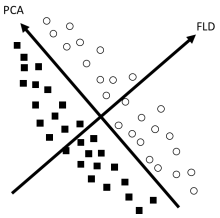
6. The slack variables are required when there is no line that can separate the classes, i.e., one cannot find a margin if the slack variables are not introduced.

7. 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.

8. One must store all training samples for the classification.

9. In the case of $k = 1$, x will belong to the circle class, since this is the closest sample. When $k = 3$, x will be classified as a square because two of the three closest samples are squares.

10. The projection directions are approximately as below.



Part 2

11.
- **Algebraic optimization**: We try to solve $\frac{\partial \epsilon}{\partial w} = 0$ algebraically, i.e., $-2(w-3) = 0 \Leftrightarrow w = 3$. If one can get a closed for expression for w as in this case, this is typically the preferable way.
- **Gradient ascent optimization**: We iterate $w_{t+1} \leftarrow w_t + \eta \frac{\partial \epsilon}{\partial w}$ (+, ascent as we seek the max), where η is the step length. If we start with a guess of $w_0 = 1$ and use $\eta = 0.25$, we get the following iteration:

$$w_1 = 1 + 0.25\,(-2(1-3)) = 2 \tag{1}$$
$$w_2 = 2 + 0.25\,(-2(2-3)) = 2.5 \tag{2}$$
$$w_3 = 2.5 + 0.25\,(-2(2.5-3)) = 2.75, \tag{3}$$

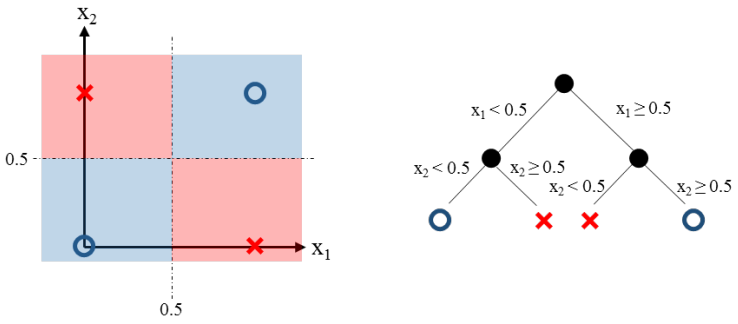
  and so on until convergence.
- **Brute force optimization**: In this approach we try different w's systematically or randomly. For example, we might try $\epsilon(1.5) = 2.75$, $\epsilon(2.5) = 4.75$, and $\epsilon(4) = 4$, and choose $w = 2.5$ as it gives the highest value for the skill.

12. The empirical risk/0-1 cost 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.

13. A possible solution is shown below. There are several other possible solutions.



14. An ”auto encoder” is a multi-layer network that is trained to reproduce the input data as output. See lecture 5.

15. The linear SVM classification function is

$$f(x; w_1, w_0) = w_1 x + w_0$$

The cost function optimized in SVM is (disregarding slack variables as the classes are separated):

$$\begin{aligned} &\min w_1^2 \\ &\text{subject to } y_i(w_1 x_i + w_0) \geq 1 \end{aligned}$$

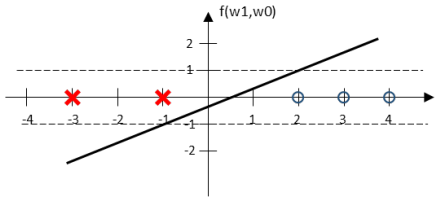
The minimum slope magnitude $w_1$is obtained when classification function fulfils the constraints at the support vectors at $x = -1$ and $x = 2$, i.e., if the label is $y = 1$ for the circles and $y = -1$ for the crosses:

$$\begin{cases} 1\,(w_1\,2 + w_0) = 1 \\ -1\,(w_1\,(-1) + w_0) = 1 \end{cases}$$

Solving for w_0 and w_1 gives the classification function

$$f(x) = \frac{2}{3}x - \frac{1}{3},$$

which is illustrated in the figure below.



Part 3

16. a) If we set a negative reward for all actions, say $r(\mathbf{s}, a) = -1$, it will model a cost for each move and promote the shortest path.

b) If the reward of any action is -1 we can set $V(1,4) = 0$ then any other location will have a lower value, and the robot will prefer the home location. The optimal move will always be up or right so we calculate these before the Q-function values for the down action.

Right:

-3	-2	-1	
-4	-3	-2	
-5	-4	-3	
-6	-5	-4	

Up:

-4	-3	-2	-1
-5	-4	-3	-2
-6	-5	-4	-3

Down:

-5	-4	-3	
-6	-5	-4	-3
-7	-6	-5	-4

c) Since the V-function contains the expected reward of taking the best action and following the optimal policy we can combine all information above to:

$V^*=$

-3	-2	-1	0
-4	-3	-2	-1
-5	-4	-3	-2
-6	-5	-4	-3

17. To train online we update the following cost function after each propagation of a training sample:

$$\mathcal{E}^m = \|\mathbf{y} - \mathbf{u}\|^2.$$

a) To get 1p all variables in the calculations must be in the drawing, including bias inputs and weights.

b) We define the weighted signal sums as:

$$h_i^h = \sum_j w_{ij}^h x_j$$

with $x_0 = 1$ for the hidden layer and

$$h_i^o = \sum_j w_{ij}^o u_j^h$$

for the output layer where u^h is the output from the hidden layer and $u_0^h = 1$. The output layers weights are updated as such

$$\Delta w_{ij}^o = -\eta \frac{\partial \mathcal{E}^m}{\partial w_{ij}^o} = 2\eta(y_i - u_i^o)\sigma'(h_i^o)u_j^h = \eta \delta_i^o u_j^h$$

and the hidden layer

$$\Delta w_{ij}^h = -\eta \frac{\partial \mathcal{E}^m}{\partial w_{ij}^h} = \eta \underbrace{\sum_{k=1}^{2} 2(y_k - u_k^o)\sigma'(h_k^o) w_{ki}^o}_{-\frac{\partial \mathcal{E}^m}{\partial u_i^o}} \underbrace{\sigma'(h_i^h)\tilde{x}_j}_{\frac{\partial u_i^o}{\partial w_{ij}}} = \eta \sigma'(h_i^h)\tilde{x}_j \sum_{k=1}^{2} (\delta_k^o w_{ki}^o)$$

and $\sigma'(h) = 1 - \sigma^2(h)$.

18. a) We have the data points:

x_1	8	6	4	5	2
x_2	1	0	-1	1	-1

We estimate the covariance matrix

$$\tilde{\mathbf{C}} = \frac{1}{N-1}(\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T = \left[\mathbf{m} = \begin{pmatrix} 5 \\ 0 \end{pmatrix}\right] = \frac{1}{5-1}\begin{pmatrix} 20 & 7 \\ 7 & 4 \end{pmatrix}.$$

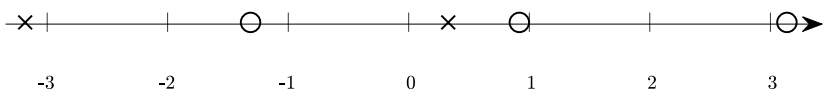
The normalized eigenvector to the largest eigenvalue is

$$\mathbf{e}_1 = \begin{pmatrix} 0.94 \\ 0.35 \end{pmatrix}$$

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

y_1	3.16	0.94	0.35	-1.29	-3.16

Plotted with the corresponded classes, it looks like this:



b) In this case we have labeled data. Looking at the figure above, we see that the first principal component does not separate the classes.

19. a) A natural representation is a string of seven zeros/ones, where a '1' represents that the corresponding item should be stolen and a '0' that the item should be left behind. The "Fitness"-function can, e.g., be chosen as the sum of the value of the things to bring if the weight is sufficiently low, and zero otherwise. Crossover can ,e.g., be implemented by copying genes before a certain position from the first parent and genes after this position are copied from the other parent. Mutation can, e.g., be implemented by changing the value at a random position from '1' to '0', or the opposite.

b) Given the choices above and assuming that the three original individuals are 1101001, 0100101 och 1010111, we have:

Individual	Weight (kg)	Value (kr)	"Fitness"
1101001	21	1900	0
0100101	14	1500	1500
1010111	20	1100	1100

The two last individuals will be copied and through crossover of these two, a new one will be created. The exact configuration of this individual depends on the definition of the crossover operator, but e.g. 0100111, having the weight 20, value 1600 and "fitness" 1600.