a) $w_0 = 0.2$, $w_1 = 1$, $w_2 = -1$

Let $\eta = 0.5$.

The input coordinates:
- For Class 1: $\{(0.08,0.72), (0.26,0.58), (0.45,0.15), (0.60,0.30)\}$
- For Class -1: $\{(0.10,1.0), (0.35,0.95), (0.70,0.65), (0.92,0.45)\}$

The initial line is $0.2 + 1*i_1 - 1*i_2 = 0$

Thus, 4 samples are misclassified, i.e.
- For Class 1: $(0.08,0.72), (0.26,0.58)$
- For Class -1: $(0.70,0.65), (0.92,0.45)$

- Choosing $(0.92, 0.45)$ from class -1 as the misclassified sample for learning.
The formula to update weight is
\[ w'_{k} = w_{k} + \text{class}(i_{k}) \cdot \eta i_{k} \]

Thus, the new weights are:
\[
\begin{align*}
  w_{1}^{0} &= 0.20 - 0.50 \times 1 = -0.3 \\
  w_{1}^{1} &= 1 - 0.5 \times 0.92 = 0.54 \\
  w_{1}^{2} &= -1 - 0.5 \times 0.45 = -1.225 
\end{align*}
\]

Thus, the new line (line 1 in the chart) is \(-0.3 + 0.54i_{1} - 1.225i_{2} = 0\)

After this update we have 4 samples misclassified - all class 1 samples are misclassified.

-Choosing (0.60, 0.30) from class 1 for learning.

The formula to update weight is
\[ w'_{k} = w_{k} + \text{class}(i_{k}) \cdot \eta i_{k} \]

Thus, the new weights are:
\[
\begin{align*}
  w_{2}^{0} &= -0.3 + 0.5 \times 1 = 0.2 \\
  w_{2}^{1} &= 0.54 + 0.5 \times 0.6 = 0.84 \\
  w_{2}^{2} &= -1.225 + 0.5 \times 0.3 = -1.075 
\end{align*}
\]

Thus, the new line (line 2 in the chart) is \(0.3 + 0.84i_{1} - 1.075i_{2} = 0\)

Then 4 samples are misclassified, i.e.

For Class 1: \((0.08,0.72), (0.26,0.58)\)
For Class -1: \((0.70,0.65), (0.92,0.45)\)

-Choosing (0.26,0.58) for class 1.

The formula to update weight is
\[ w'_{k} = w_{k} + \text{class}(i_{k}) \cdot \eta i_{k} \]

Thus, the new weights are:
\[
\begin{align*}
  w_{3}^{0} &= 0.2 + 0.5 \times 1 = 0.7 \\
  w_{3}^{1} &= 0.84 + 0.5 \times 0.26 = 0.97 \\
  w_{3}^{2} &= -1.075 + 0.5 \times 0.58 = -0.785 
\end{align*}
\]

Thus, the new line (line 3 in the chart) is \(0.7 + 0.97i_{1} - 0.785i_{2} = 0\)

After this update we have 4 samples misclassified. All class -1 samples are misclassified.

-Choosing (0.70,0.65) from class -1 for learning.

The formula to update weight is
\[ w'_{k} = w_{k} + \text{class}(i_{k}) \cdot \eta i_{k} \]

Thus, the new weights are:
\[ w^4_0 = 0.7 - 0.5 \times 1 = 0.2 \\
\[w^4_1 = 0.97 - 0.5 \times 0.7 = 0.62 \\
\[w^4_2 = -0.785 - 0.5 \times 0.65 = -1.11 \\

Then the new line (line 4 in the chart) is \(0.2 + 0.62i_1 - 1.11i_2 = 0\)

Thus, 3 samples are misclassified, i.e.

For Class 1: (0.08,0.72), (0.26,0.58)
For Class -1: (0.92,0.45)

- Choosing (0.26,0.58) from class 1 for learning.

The formula to update weight is
\[ w'_k = w_k + \text{class}(i_k) \times \eta \times i_k \]

Thus, the new weights are:

\[ w'^4_0 = 0.2 + 0.5 \times 1 = 0.7 \\
\[w'^4_1 = 0.62 + 0.5 \times 0.26 = 0.75 \\
\[w'^4_2 = -1.11 + 0.5 \times 0.58 = -0.82 \\

Then the new line (line 5 in the chart) is \(0.7 + 0.75i_1 - 0.82i_2 = 0\)

Thus, 2 samples are misclassified, i.e.

For Class 1: (0.60,0.30)
For Class -1: (0.10,1.00)

-Choosing (0.10, 1.00) for class -1.

The formula to update weight is
\[ w'_k = w_k + \text{class}(i_k) \times \eta \times i_k \]

Thus, the new weights are:

\[ w'^5_0 = 0.7 - 0.5 \times 1 = 0.2 \\
\[w'^5_1 = 0.75 - 0.5 \times 0.1 = 0.70 \\
\[w'^5_2 = -0.82 - 0.5 \times 1 = -1.32 \\

Thus, the new line (line 6 in the chart) is \(0.2 + 0.70i_1 - 1.32i_2 = 0\)

Thus, 3 samples are misclassified, i.e.

For Class 1: (0.08,0.72), (0.26,0.58)
For Class -1: (0.92,0.45)

b) The separating line on the chart separates the classes perfectly. Its formula is
\[ 1 - 0.8i_1 + 1i_2 = 0 \]
Thus, the weights are \(w_0 = 1, w_1 = -0.8\) and \(w_2 = 1\).

c) The input coordinates in this case are:
For Class 1: \{0.08, 0.26, 0.45, 0.60\}
For Class -1: \{0.10, 0.35, 0.70, 0.92\}

The above diagram shows the samples in one-dimensional case.

It’s clear from the diagram that the best results will have two samples to be misclassified, i.e., the resulting error will be $2/8 = 0.25$.

If, for instance, we classify all the points left to 0.65 to be class 1 and class -1 to the right, then the minimum error will be reached, and the separating point could have the following formula:

$$0.65 - 1*i_{1} = 0$$

Thus, $w_0 = 0.65$ and $w_1 = -1$ will do the separation with the minimal error.

Question 2:

The input coordinates:
For Class 1: \{(0.08,0.72), (0.20,0.50), (0.24,0.30), (0.35,0.35), (0.45,0.50)\}
For Class -1: \{(0.02,0.48), (0.10,1.00), (0.36,0.75), (0.52,0.24), (0.70,0.65), (0.80,0.26), (0.92,0.45)\}

a)

The minimum error in this case for one perceptron case would be 2 misclassified samples, i.e., an error value of $2/12 = 0.167$. 
For the line shown in the diagram above, two samples for class -1: (0.02,0.48) and (0.52,0.24) are missedclassified. This line has the following formula:

\[ 1 - 1 \cdot i_1 - 1 \cdot i_2 = 0 \]

Thus, \( w_0 = 1, w_1 = -1 \) and \( w_2 = -1 \) will do the separation with the minimal error.

b)

If everything inside of this triangle (i.e. 3 lines intersections) will be classified as class -1 and everything outside of the triangle as class -1, then the perfect classification will be achieved for these data. So we need 3 perceptrons to compute these classification functions:

The formula for line 1 would be:

\[-0.2 + 1 \cdot i_1 + 0.2 \cdot i_2 = 0\]

Thus, \( w_{1,0} = -0.2, w_{1,1} = 1 \) and \( w_{1,2} = 0.2 \) for perceptron 1.

The formula for line 2 would be:

\[1 - 1 \cdot i_1 - 1 \cdot i_2 = 0\]

Thus, \( w_{2,0} = 1, w_{2,1} = -1 \) and \( w_{2,2} = -1 \) for perceptron 2.

The formula for line 3 would be:

\[0.2 - 1 \cdot i_1 + 0.60 \cdot i_2 = 0\]

Thus, \( w_{3,0} = 0.2, w_{3,1} = -1 \) and \( w_{3,2} = 0.60 \) for perceptron 3.

Now we just need to apply the “and” function to these three results. For this purpose, let us introduce perceptron 4, which is the only unit in the second layer of our network. Like the other perceptrons, this neuron receives an offset (“dummy”) input of 1, and it also receives the three outputs from units 1 to 3, for a total of 4 inputs. If we set \( w_{4,0} = -2.5, w_{4,1} = 1, w_{4,2} = 1 \) and \( w_{4,3} = 1 \), then the output is 1 if and only if all first-layer perceptrons output 1.
Therefore, the structure of the multi-layer perceptron net is the following, with the weights set as described above:

Question 3:

(a) If the new dimensions actually contain information that is relevant to the classification task and is not already contained in the initial dimensions, then the perceptron is more likely to learn the classification task perfectly. Just imagine the XOR task in a two-dimensional input space, which we know cannot be learned perfectly by any perceptron. It consists of the following exemplars:

\begin{align*}
    x_1 &= (0, 0), \quad d_1 = -1 \\
    x_2 &= (1, 0), \quad d_2 = 1 \\
    x_3 &= (0, 1), \quad d_3 = 1 \\
    x_4 &= (1, 1), \quad d_4 = -1
\end{align*}

Now let us add a relevant third dimension:

\begin{align*}
    x_1 &= (0, 0, 0.2), \quad d_1 = -1 \\
    x_2 &= (1, 0, 0.8), \quad d_2 = 1 \\
    x_3 &= (0, 1, 0.7), \quad d_3 = 1 \\
    x_4 &= (1, 1, 0.3), \quad d_4 = -1
\end{align*}
Now our perceptron can simply divide this three-dimensional input space with a two-dimensional plane whose position in the new dimension is, for example, a constant 0.5, to compute the desired function perfectly. Of course this is an extreme example – now the third dimension alone would be sufficient to do the classification task. However, as we also saw in Question 1c, the more dimensions our input space and the dividing plane (line) have, the better is our chance to find a perfect linear separation, as long as the additional dimensions are relevant for the current classification task.

(b) No, there is no minimum number of dimensions that would guarantee linear separability. Just look at the XOR task again:

\begin{align*}
  x_1 &= (0, 0), d_1 = -1 \\
  x_2 &= (1, 0), d_2 = 1 \\
  x_3 &= (0, 1), d_3 = 1 \\
  x_4 &= (1, 1), d_4 = -1
\end{align*}

Now if we add dimensions that are irrelevant to the task, our perceptron still cannot solve the higher-dimensional classification task:

\begin{align*}
  x_1 &= (0, 0, 0), d_1 = -1 \\
  x_2 &= (1, 0, 0), d_2 = 1 \\
  x_3 &= (0, 1, 0), d_3 = 1 \\
  x_4 &= (1, 1, 0), d_4 = -1
\end{align*}

\begin{align*}
  x_1 &= (0, 0, 0, 1), d_1 = -1 \\
  x_2 &= (1, 0, 0, 1), d_2 = 1 \\
  x_3 &= (0, 1, 0, 1), d_3 = 1 \\
  x_4 &= (1, 1, 0, 1), d_4 = -1
\end{align*}

… and so on. We can add any number of dimensions this way without creating linear separability.