Exercises

E4.1 Consider the classification problem defined below:

$$\left\{\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_1 = 1\right\} \left\{\mathbf{p}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_2 = 1\right\} \left\{\mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_3 = 1\right\} \left\{\mathbf{p}_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_4 = 0\right\}$$

$$\left\{\mathbf{p}_5 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t_5 = 0\right\}.$$

i. Draw a diagram of the single-neuron perceptron you would use to solve this problem. How many inputs are required?

ii. Draw a graph of the data points, labeled according to their targets. Is this problem solvable with the network you defined in part (i)? Why or why not?

E4.2 Consider the classification problem defined below.

$$\left\{\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_1 = 1\right\} \left\{\mathbf{p}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, t_2 = 1\right\} \left\{\mathbf{p}_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_3 = 0\right\} \left\{\mathbf{p}_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_4 = 0\right\}.$$

i. Design a single-neuron perceptron to solve this problem. Design the network graphically, by choosing weight vectors that are orthogonal to the decision boundaries.

ii. Test your solution with all four input vectors.

» 2 + 2
ans =
4

iii. Classify the following input vectors with your solution. You can either perform the calculations manually or with MATLAB.

$$\mathbf{p}_5 = \begin{bmatrix} -2 \\ 0 \end{bmatrix} \qquad \mathbf{p}_6 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \mathbf{p}_7 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathbf{p}_8 = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

iv. Which of the vectors in part (iii) will always be classified the same way, regardless of the solution values for $\mathbf{W}$ and b? Which may vary depending on the solution? Why?

E4.3 Solve the classification problem in Exercise E4.2 by solving inequalities (as in Problem P4.2), and repeat parts (ii) and (iii) with the new solution. (The solution is more difficult than Problem P4.2, since you can't isolate the weights and biases in a pairwise manner.)

E4.4 Solve the classification problem in Exercise E4.2 by applying the perceptron rule to the following initial parameters, and repeat parts (ii) and (iii) with the new solution.

$$\mathbf{W}(0) = \begin{bmatrix} 0 & 0 \end{bmatrix} \qquad b(0) = 0$$

E4.5 Prove mathematically (not graphically) that the following problem is unsolvable for a two-input/single-neuron perceptron.

$$\left\{\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_1 = 1\right\} \left\{\mathbf{p}_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, t_2 = 0\right\} \left\{\mathbf{p}_3 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_3 = 1\right\} \left\{\mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_4 = 0\right\}$$

(Hint: start by rewriting the input/target requirements as inequalities that constrain the weight and bias values.)

E4.6 We have four categories of vectors.

$$\text{Category I: } \left\{\begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}\right\}, \text{ Category II: } \left\{\begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right\}$$

$$\text{Category III: } \left\{\begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}\right\}, \text{ Category IV: } \left\{\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}\right\}$$

i. Design a two-neuron perceptron network (single layer) to recognize these four categories of vectors. Sketch the decision boundaries.

ii. Draw the network diagram.

iii. Suppose the following vector is to be added to Category I.

$$\begin{bmatrix} -1 \\ -3 \end{bmatrix}$$

Perform one iteration of the perceptron learning rule with this vector. (Start with the weights you determined in part i.) Draw the new decision boundaries.

E4.7 We have two categories of vectors. Category I consists of

$$\left\{\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right\}.$$

Category II consists of

$$\left\{ \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} -2 \\ 0 \end{bmatrix} \right\}.$$

i. Design a single-neuron perceptron network to recognize these two categories of vectors.

ii. Draw the network diagram.

iii. Sketch the decision boundary.

iv. If we add the following vector to Category I, will your network classify it correctly? Demonstrate by computing the network response.

$$\begin{bmatrix} -3 \\ 0 \end{bmatrix}$$

v. Can your weight matrix and bias be modified so your network can classify this new vector correctly (while continuing to classify the other vectors correctly)? Explain.

E4.8 We want to train a perceptron network with the following training set:

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, t_1 = 0 \right\} \left\{ \mathbf{p}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_2 = 0 \right\} \left\{ \mathbf{p}_3 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_3 = 1 \right\}.$$

The initial weight matrix and bias are

$$\mathbf{W}(0) = \begin{bmatrix} 1 & 0 \end{bmatrix},\ b(0) = 0.5\,.$$

i. Plot the initial decision boundary, weight vector and input patterns. Which patterns are correctly classified using the initial weight and bias?

ii. Train the network with the perceptron rule. Present each input vector once, in the order shown.

iii. Plot the final decision boundary, and demonstrate graphically which patterns are correctly classified.

iv. Will the perceptron rule (given enough iterations) always learn to correctly classify the patterns in this training set, no matter what initial weights we use? Explain.

E4.9 We want to train a perceptron network using the following training set:

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\0\end{bmatrix}, t_1 = 0\right\}\left\{\mathbf{p}_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\}\left\{\mathbf{p}_3 = \begin{bmatrix}1\\2\end{bmatrix}, t_3 = 1\right\},$$

starting from the initial conditions

$$\mathbf{W}(0) = \begin{bmatrix}0 & 1\end{bmatrix}, \; b(0) = \begin{bmatrix}1\end{bmatrix}.$$

i. Sketch the initial decision boundary, and show the weight vector and the three training input vectors, $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$. Indicate the class of each input vector, and show which ones are correctly classified by the initial decision boundary.

ii. Present the input $\mathbf{p}_1$ to the network, and perform one iteration of the perceptron learning rule.

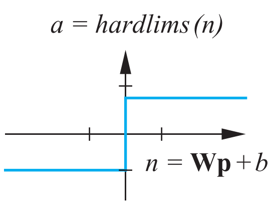
iii. Sketch the new decision boundary and weight vector, and again indicate which of the three input vectors are correctly classified.

iv. Present the input $\mathbf{p}_2$ to the network, and perform one more iteration of the perceptron learning rule.

v. Sketch the new decision boundary and weight vector, and again indicate which of the three input vectors are correctly classified.

vi. If you continued to use the perceptron learning rule, and presented all of the patterns many times, would the network eventually learn to correctly classify the patterns? Explain your answer. (This part does not require any calculations.)

E4.10 The symmetric hard limit function is sometimes used in perceptron networks, instead of the hard limit function. Target values are then taken from the set [-1, 1] instead of [0, 1].



i. Write a simple expression that maps numbers in the ordered set [0, 1] into the ordered set [-1, 1]. Write the expression that performs the inverse mapping.

ii. Consider two single-neuron perceptrons with the same weight and bias values. The first network uses the hard limit function ([0, 1] values), and the second network uses the symmetric hard limit function. If the two networks are given the same input $\mathbf{p}$, and updated with the perceptron learning rule, will their weights continue to have the same value?

iii. If the changes to the weights of the two neurons are different, how do they differ? Why?

iv. Given initial weight and bias values for a standard hard limit perceptron, create a method for initializing a symmetric hard limit perceptron so that the two neurons will always respond identically when trained on identical data.

E4.11 The vectors in the ordered set defined below were obtained by measuring the weight and ear lengths of toy rabbits and bears in the Fuzzy Wuzzy Animal Factory. The target values indicate whether the respective input vector was taken from a rabbit (0) or a bear (1). The first element of the input vector is the weight of the toy, and the second element is the ear length.

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\4\end{bmatrix}, t_1 = 0\right\} \left\{\mathbf{p}_2 = \begin{bmatrix}1\\5\end{bmatrix}, t_2 = 0\right\} \left\{\mathbf{p}_3 = \begin{bmatrix}2\\4\end{bmatrix}, t_3 = 0\right\} \left\{\mathbf{p}_4 = \begin{bmatrix}2\\5\end{bmatrix}, t_4 = 0\right\}$$

$$\left\{\mathbf{p}_5 = \begin{bmatrix}3\\1\end{bmatrix}, t_5 = 1\right\} \left\{\mathbf{p}_6 = \begin{bmatrix}3\\2\end{bmatrix}, t_6 = 1\right\} \left\{\mathbf{p}_7 = \begin{bmatrix}4\\1\end{bmatrix}, t_7 = 1\right\} \left\{\mathbf{p}_8 = \begin{bmatrix}4\\2\end{bmatrix}, t_8 = 1\right\}$$

i. Use MATLAB to initialize and train a network to solve this "practical" problem.

ii. Use MATLAB to test the resulting weight and bias values against the input vectors.

iii. Add input vectors to the training set to ensure that the decision boundary of any solution will not intersect one of the original input vectors (i.e., to ensure only robust solutions are found). Then retrain the network. Your method for adding the input vectors should be general purpose (not designed specifically for this problem).

E4.12 Consider again the four-category classification problem described in Problems P4.3 and P4.5. Suppose that we change the input vector $\mathbf{p}_3$ to

$$\mathbf{p}_3 = \begin{bmatrix}2\\2\end{bmatrix}.$$

i. Is the problem still linearly separable? Demonstrate your answer graphically.

ii. Use MATLAB to initialize and train a network to solve this problem. Explain your results.

iii. If $\mathbf{p}_3$ is changed to

$$\mathbf{p}_3 = \begin{bmatrix}2\\1.5\end{bmatrix}$$

is the problem linearly separable?

iv. With the $\mathbf{p}_3$ from (iii), use MATLAB to initialize and train a network to solve this problem. Explain your results.

E4.13 One variation of the perceptron learning rule is

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{e} \mathbf{p}^T$$

$$\mathbf{b}^{new} = \mathbf{b}^{old} + \alpha \mathbf{e}$$

where α is called the learning rate. Prove convergence of this algorithm. Does the proof require a limit on the learning rate? Explain.