

[Studenti](#)[Ricerca](#)[Ateneo](#)[Servizi online](#)[Intranet](#)[Aulaweb](#)

[Dibris](#) ► [My courses](#) ► [Robotics Engineering \(Scuola Politecnica\)](#) ► [Anno Accademico 2013/14](#) ► [65862-1314](#) ► [Lab assignments](#) ► ...ent 2: Adaline and non-linearly separable problems

You are logged in as **Muhammad Farhan Ahmed** ([Logout](#))

## Navigation



[Dibris](#)

■ [My home](#)

[Dibris](#)

[My profile](#)

[Current course](#)

[65862-1314](#)

[Participants](#)

[Badges](#)

[General](#)

[Activity dates and topics](#)

[Notes and slides](#)

[Lab assignments](#)

[Lab submission guidelines](#)

[Lab assignment 1: one-layer perceptrons](#)

[...ent 2: Adaline and non-linearly separable problems](#)

[Feedback on](#)

## Lab assignment 2: Adaline and non-linearly separable problems

- Task 1: Adaline
- Task 2: Handwritten character recognition
- Task 3 (optional): Classifying non-linearly-separable data sets

Describe everything in a report.

### Task 1: Adaline

Write a program module (a function) implementing Widrow's Adaline, learning with the delta rule. Since learning is by pattern, i.e., at each input pattern the weights change, it is advisable that after the last iteration a final test is done on the whole training set, to assess the performance of the final set of weights.

The program will compute some performance indexes: number of classification errors, norm of the gradient, mean squared error. It will have a stopping criterion for each of these, so that it will finish training if the total classification error (over all the training set) is zero, or if the mse is under a certain threshold, or if the norm of the gradient is under a certain threshold (necessary minimum condition). The program will also stop after a given maximum number of "training epochs", i.e., passes over the whole training set.

It might be useful to have a printout of the computed quantities, to see what's going on, but not at each iteration (they would be too many prints that would slow down the algorithm).

In Matlab, as well as C, you can use a function named printf. In C++ you can use the same function, or you can use the cout stream object.

The program should have the following structure:

```
initialize overall parameters (e.g., learning rate eta)
repeat "maxepoch" times:
  set all performance indexes to zero
  repeat for each input pattern:
    compute deltaW
```

[course workload](#)[Lab assignment](#)

3:

[Prototype-based classifiers](#)[Lab assignment](#)

4: Training

multi-layer

networks

[Lab assignment](#)

5:

[Unsupervised](#)

learning

[Lab resources](#)[General resources](#)[My courses](#)[Courses](#)

## Administration

[Course administration](#)[My profile settings](#)

```
    update performance indexes
  finalize performance indexes
  check stopping conditions and, if appropriate, stop iterating
perform final test and exit, returning weights and test results
```

Tip: Most quantities are more significant if they are made independent of the number of inputs and weights. For instance, the norm of the gradient should be actually the average (over all training data) of the root mean square weights (over all inputs).

### Task 2: Handwritten character recognition

Download the Semeion digit recognition data set [as a zip archive](#) or [as a tar/gz archive](#). Also read the [readme file](#) and download the [ready-made input script](#).

Use your Adaline program to solve the handwritten digit recognition problem. You will need an array of Adalines, each one trained to recognize a digit.

While setting up your problem, it is advisable to try with a subset of the data first (e.g., the first 100 or 200 patterns).

You will find that the basic training procedure must be refined. For instance, the size of the learning rate *eta* will have to be carefully tuned to balance between being accurate but too slow and being fast but too inaccurate (over a certain value the weights may even diverge). Also, to ensure convergence (reducing stochastic oscillations), *eta* could be gradually lowered.

Is a good *eta* still good when switching from 100 or 200 to all 1593 patterns?

Is the training set linearly separable? Are you sure?

Compare the behaviour and performance of the Adaline with that of the perceptron: run analogous experiments with both neuron models on the digit recognition problem and report on the results. In particular, the number of errors could be different: verify and comment.

### Task 3 (optional): Classifying non-linearly-separable data sets

Build a simple dataset for the XOR problem, whose truth table is:

x1	x2	t
0	0	0
1	0	1
0	1	1
1	1	0

This problem is not linearly separable. Find ways to solve it with linear threshold logic units, a.k.a. McCulloch-Pitts neurons, a.k.a. binary-input, binary-output perceptrons.

Experiment with the following strategies:

**Non-linear separating surface** - Change the function that computes  $r$  from a linear one (weighted sum or dot product) to a suitable, non-linear one.

**Pre-processing of features** - Add one or more columns to the training set. The content of the columns for each pattern should be some value computed on the already present columns  $x_1$  and  $x_2$ . You may design these additional values so as to make the data set linearly separable, but these should be strictly functions of  $x_1$  and  $x_2$ ; adding more information (e.g., the target value) is not allowed!


Try both linear (linear combinations) and non-linear (general, e.g., polynomial) functions.

**Learning features** - Instead of using  $x_1$  and  $x_2$ , use the output of two other suitable logical functions of  $x_1$  and  $x_2$ . The logical function "XOR" can be implemented as a composition of other logical functions. Implement these as linear threshold logic units, train them with the perceptron or adaline methods, and feed the array of their outputs to a final decision stage, an output linear threshold unit.

Comment on what you find about these strategies.

UPLOAD BELOW YOUR CODE, DATA, AND REPORT

### Submission status

Submission status	Submitted for grading
Grading status	Not graded
Due date	Wednesday, 30 October 2013, 11:55 PM
Time remaining	Assignment was submitted 2 days 23 hours early
Last modified	Monday, 28 October 2013, 12:26 AM
File submissions	 <a href="#">LAB@2_Ahmed.rar</a>