Introduction to Artificial Intelligence
Internet of Things and Data Analytics
Final project

Hector Garcia de Marina
hgarciad@ucm.es

March 21, 2022

Chapter 1

Summary

The final project consists of two separated exercises. The first one deals with handwritting recognition. The exercise covers the algorithms of Neural Networks, K-nearest neighbors, and Support Vector Machines. The second exercise deals with planning the route and guiding a robot in an environment with obstacles. The exercise covers the algorithms of Genetic Algorithms and Guiding Vector Fields.

1.1 Scores

In order to pass the course, the two exercises are mandatory. If the sent report is satisfactory, then (and only then) the Quizzes and the Assigments during the course will be considered to increase the final score of the student. The scores will be reported in a scale out of 10, being 5 the minimum score to pass, and 10 the maximum score.

1.2 Report

The two exercises must be sent in two different reports. In particular, as a Jupyter notebook. It must be very clear at the beguinning of the report the names of the students. Please, note that **only code will not be enough**, and discussions and comments on the methodology and results are necessary for passing.

1.3 Deadline

There are six sessions where the students can ask and interact with the teacher on the exercises. After that, the exercises must be sent before the **6th of May**.

Chapter 2

Handwriting recognition

From the database http://yann.lecun.com/exdb/mnist/ you can find a test set of 28×28 pixel images with handwritten numbers from 0 to 9 in it.

You have to write and verify the following programs:

- Code a Neural Network with PyTorch. Try different architectures, i.e., different number of neurons, hidden layers, loss functions, etc, and comment the differences on the results.
- Code a K-nearest neighbors algorithm. Consider only the numbers $0, 1$, and 2. Use part of the database as a known points/numbers, and the other part as for testing the algorithm.
- Code a Support Vector Machine. Consider only the numbers $0, 1$, and 2. Use part of the database as known points/numbers, and the other part for testing the algorithm. Try different kernels and comment the differences on the results.

For the K-nearest neighbors and Support Vector Machines algorithms, it is enough if you use 500 samples for training/known points, and another different 100 samples for testing the algorithm.

Chapter 3

Path planning and guiding of a mobile robot

Given a map of 100×100 length units. We fill it with $n > 1$ rectangular obstacles R_i.

$$R_i := \{{}^1R_i, {}^2R_i, {}^3R_i, {}^4R_i\}, \tag{3.1}$$

where ${}^jR_i \in \mathbb{R}^2$ are the four corners of the rectangle.

We have a robot starting from an arbitrary position $p_0 \in \mathbb{R}^2$, and we need to guide it to a target position $p_T \in \mathbb{R}^2$ within the map. Check that both positions do not lie within a rectangle/obstacle.

You have to write and verify the following programs:

- Code a function that generates a map of 100×100 with n rectangular obstacles. Use Matplotlib to draw it.
- Code a Genetic Algorithm that generates $s \in \mathbb{N}$ segments connecting p_0 and p_T, and these segments do not lie within any rectangular obstacle. You can define a segment with a starting and ending point, for example $s_j := \{start_x, start_y, end_x, end_y\}$.
- Given the series of segments, code a simulation where the robot follows the segments employing the Guiding Vector Field algorithm to reach the destination p_T.

In the genetic algorithm, you can use the chromosomes to codify the information of the s segments. For example, chromosome $:= \{s_{1_x}, s_{1_y}, \ldots, s_{s_x}, s_{s_y}\}$. Note that the ending point of a segment is the starting one for the next segment. The fitness function can be the summation of the length all the segments, plus the segment form by p_0 and s_1 and by s_n and p_T. If one the segments *hits* an obstacle, then you can add a penatly of 9999 to the fitness function.

For the simulation of the robot's motion, we are going to consider noise. For example, given a time step ΔT, then the position of the robot must be updated using the following:

$$p(k+1) = p(k) + directionOfMotion \times rand \times \Delta T, \tag{3.2}$$

where $rand \in \mathbb{R}^2$ is a random vector with norm one, and k is the iteration in the simulation loop.