Distributed Autonomous Systems

Course Project

The project consists in two main tasks. The first one involves a data analytics application while the second one deals with the control for multi-robot systems in ROS 2.

Task 1: Distributed Classification via Neural Networks

Task 1 concerns the classification of a set of images. Suppose to have a team of N agents. Each agent i can access only to a private set containing $m_i \in \mathbb{N}$ images $\mathcal{D}^j \in \mathbb{R}^d$ with associated label y^j coming from a given dataset. The dataset is split in a training set $\{\mathcal{D}^j, y^j\}_{j=1}^{m_{\text{train}}}$ and a test set $\{\mathcal{D}^h, y^h\}_{h=1}^{m_{\text{test}}}$.
Two possible datasets are available:

(a) handwritten digits (`mnist`);

(b) Zalando's article images (`fashion_mnist`).

The assignment of datasets will depend on the group number. Odd groups will be assigned option (a), while even groups will be assigned option (b). For instance, Group 23 must use the `mnist` dataset, and Group 12 must use the `fashion_mnist` dataset.

Task 1.1 – Distributed Optimization

1. Implement the *Gradient Tracking* algorithm to solve a consensus optimization problem in the form

$$\min_u \sum_{i=1}^{N} J_i(u)$$

   where J_i is a quadratic function.

2. Run a set of simulations to test the effectiveness of the implementation. Moreover, provide a set of solutions that includes different weighted graph patterns (e.g., cycle, path, star) whose weights are determined by the Metropolis-Hastings method. Finally, for each simulation, plot the evolution of the cost function and of the norm of the gradient of the cost function across the iterations.

Task 1.2 – Centralized Training

1. Prepare the dataset for neural network training. Select a category (e.g., the label "Sandal")

   (a) assign the label 1 to all the images belonging to the selected category;

   (b) assign the label 0 to all other images, not belonging to the selected category;

   (c) take only a reduced subset of images.

2. Implement the multi-sample neural network, extending the one-sample example presented during the lectures, with:

   - Sigmoid function as activation function $\sigma(\cdot)$;

- Binary Cross-Entropy (BCE) as loss function $\ell(\cdot)$.

Note. Students can freely choose other activation and/or loss functions in their implementations. However, it is important to note that the evaluation will primarily focus on the distributed implementation of the algorithm.

Task 1.3 – Distributed Training

1. Split (randomly) the entire training set in N subsets, one for each agent i.
2. Implement a distributed algorithm to train a neural network based on the *Gradient Tracking* (you are allowed to extend the code provided during the lectures)
3. Generate a set of simulations showing the convergence of the distributed algorithm to a stationary point of the optimization problem. Moreover, plot the evolution of the cost function and of the norm of the gradient of the cost function across the iterations.
4. Test different dataset sizes (start with a small number of samples)
5. Evaluate the quality of the obtained solution by computing its accuracy (say computed by agent 0) on the test set. That is, compute the percentage of success of the following test, for $j = 1, \dots, m_{\text{test}}$

$$\hat{y}^j = \phi(\mathbf{u}^\star, \mathcal{D}^j) = \begin{cases} 1 & \text{if } (\mathbf{u}^\star)^\top \mathcal{D}^j \geq 0 \\ 0 & \text{if } (\mathbf{u}^\star)^\top \mathcal{D}^j < 0. \end{cases}$$

   The classifier succeeds if $\hat{y}^j = y^j$.

Hints:

1. **Important:** you are allowed to use the files provided during the exercise lectures
2. Reshape and normalize the samples so that $\mathcal{D}^j \in [0,1]^{784}$
3. The dataset can be imported from the Keras Python Library (`from keras.datasets import mnist, fashion_mnist`)

Task 2: Formation Control

Consider a team of N robots. We denote the position of robot $i \in \{1, \dots, N\}$ at time $t \geq 0$ with $x_i(t) \in \mathbb{R}^3$, and with $p_i^k \in \mathbb{R}^3$ its discretized version at time $k \in \mathbb{N}$. Hence, we represent with $x(t) \in \mathbb{R}^{3N}$ and $p^k \in \mathbb{R}^{3N}$ respectively the stack vector of the continuous and the discrete positions.

Task 2.1 – Problem Set-up

1. Implement in ROS 2 a discrete-time version of the *Formation Control* law based on potential. Choose as potential function

$$V_{ij}(x) = \frac{1}{4}\Big(\|x_i - x_j\|^2 - d_{ij}^2\Big)^2,$$

with $d_{ij} \in \mathbb{R}$ the assigned distances between two robots. As a consequence, for each robot i:

$$\dot{x}_i(t) = f_i(x(t)) = -\sum_{j \in \mathcal{N}_i} \Big(\|x_i(t) - x_j(t)\|^2 - d_{ij}^2\Big)(x_i(t) - x_j(t))$$
$$p_i^{k+1} = p_i^k + \Delta \cdot f_i(p^k)$$

where $\Delta > 0$ is the sampling period (i.e., the node frequency in ROS 2). Start from the Python files and the ROS 2 consensus package provided during the lectures.

2. Run a set of simulations choosing different formation patterns (e.g., letters, numbers, polygons) and a different number of agents, providing an animated visualization of the team behavior (you can also use the RViz template provided during the exercise lectures).

Task 2.2 – Collision Avoidance

1. Implement a modified version of the *Formation Control* that includes collision avoidance barrier functions. A candidate barrier function could be

$$V_{ij}(x) = -\log(\|x_i(t) - x_j(t)\|^2).$$

2. Run a set of simulations showing the effectiveness of the barrier functions.

Task 2.3 – Moving Formation and Leader(s) control

1. Declare one agent (or more) as leader of the formation and implement a control law (e.g., a proportional controller) to steer the formation toward a target position.

2. Run a set of simulations choosing different target positions.

Task 2.4 – (Optional) Obstacle Avoidance

1. From Task 2.2, implement an obstacle avoidance algorithm;

2. Run a set of simulations validating the algorithm.

Notes

1. Each group must be composed of at most 3 students.
2. Each group must attend at least 2 meetings with the tutor.
3. All the emails for the project support must have the subject:
   "[DAS2023]-Group X: *support request*".

   Moreover, all the group members, the tutor, and the professors must always be included in the emails.
4. The project report must be written in LaTeX and must follow the main structure of the provided template.
5. The final submission **deadline** is **one** week before the exam date.
6. **Final submission**: one member of each group must send an email with subject "[DAS2023]-Group X: Submission", with attached a link to a OneDrive folder, containing:
   - `README.txt`
   - `report_group_XX.pdf`
   - `report` – a folder containing the LaTeX code and a `figs` folder (if any)
   - `task_1` – a folder containing the code relative to Task 1, including `README.txt`
   - `task_2` – a folder containing the code relative to Task 2, including `README.txt`
7. Any other information and material necessary for the project development will be given during project "meetings".