UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Distributed Autonomous Systems

DISTRIBUTED CLASSIFICATION VIA NEURAL NETWORKS AND FORMATION CONTROL

Professors:
Giuseppe Notarstefano
Ivano Notarnicola

Students:
Simone Cenerini
Giulia Cutini
Riccardo Paolini

Academic year 2022/2023

Abstract

Contents

Introduction

Motivations

Contributions

Chapter 1

Distributed classification via Neural Network

In the first task we were asked to correctly classify a set of grayscale images. The dataset, downloaded from the Keras `mnist`, collects images of handwritten digits of 28×28 pixels each one, an example in figure 1.1. In order to perform our task we were asked to implement a distributed classification: given a predefined number of agents running each one a neural network with the same structure, we were asked to implement a Gradient Tracking algorithm to ensure consensus of the connection weights of the several neural networks runned separately by different agents.

1.1 Initial setup

The first step was to prepare the dataset by a reshape and a normalization of the images. We have flattened the 28×28 pixels grayscale images in order to obtain a $[784, 1]$ column vector and we have normalized the pixels' intensities by dividing each intensity by 255 in order to avoid a saturation of the activation function.

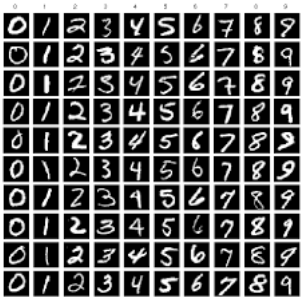

Figure 1.1: Example of images from the `mnist` dataset

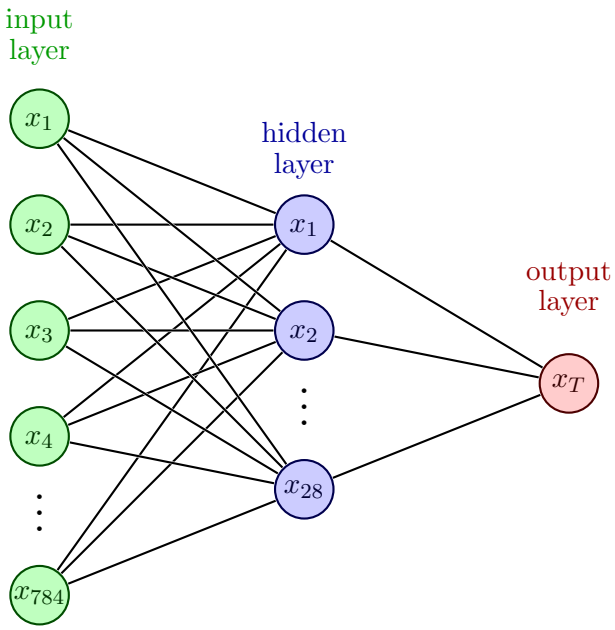

Figure 1.2: Scheme of the neural network for image classification

The classification we were asked to perform was a binary one: we have choose one among the ten digits (named as `target`) and assigned to it the label 1, while, to all the other images, we have assigned the label 0, in the following way:

$$y_i = \begin{cases} 1, & \text{if label} = \texttt{target} \\ 0, & \text{otherwise} \end{cases} \tag{1.1}$$

As a consequence, we have performed a **balancing of the dataset**: in order to efficiently train our neural network we have equalized the representation of both the classes in the dataset (50% of both since we have two classes). In order to do that we have undersampled the number of samples in the majority class.

1.2 Neural Network structure

The structure we have choose to develop for the neural network is a tapered one, represented in figure 1.2. The first layer is composed by 784 neurons, that is the dimension of the flattened image we give as input in the network. In this way each neurons of the first layer gets a single pixel as input. The second layer, the hidden one, has 28 neurons since an higher number of neurons was useless in order to accomplish our task, while it increases a lot the computational effort required to train the network. The last layer has

just one single neuron since we are asked to perform a binary classification, and so just a number between 0 and 1 is required as prediction.

Chapter 2

Formation Control

In the second task we were asked to implement in ROS2 a discrete version of a Formation Control law for a team of N robots.

2.1 Distance-based formation

The aim of the task was to obtain a desired geometric formation among a group of N autonomous agents, by acting on the relative positions of agents in order to achieve a specified formation pattern. The desired formation can be encoded in terms of an undirected graph, called *formation graph*, whose set of vertices is indexed by the team members $\mathcal{N} = \{1, \cdots, N\}$, and whose set of edges $E = \{(i, j) \in \mathcal{N} \times \mathcal{N} | \, j \in \mathcal{N}_i\}$ contains pairs of vertices. Each edge $(i, j) \in E$ is assigned a scalar parameter $d_{ij} = d_{ji} > 0$, representing the distance at which agents i, j should converge to.

An example could be a desired formation with hexagon shape with the following *distances matrix*:

$$d = \begin{bmatrix} 0 & L & 0 & D & H & L \\ L & 0 & L & 0 & D & 0 \\ 0 & L & 0 & L & 0 & D \\ D & 0 & L & 0 & L & 0 \\ H & D & 0 & L & 0 & L \\ L & 0 & D & 0 & L & 0 \end{bmatrix} \tag{2.1}$$

The desired distance value serve as reference value for the control law. The control law computes control signals for each agent based on its current position and the position of the neighboring agents, aiming to drive the agents towards the desired formation. The neighbors of a certain agent i can be extracted from the i-th row of the distances matrix, by taking the indexes of the elements with $d_j > 0$.

One way to approach distance-based formation control involves the usage of potential functions, similar to the following one:

$$V_{ij}(x) = \frac{1}{4}\left(\|x_i - x_j\|^2 - d_{ij}^2\right)^2 \tag{2.2}$$

This kind of potential function represents the energy associated with the relative positions of the agents. By minimizing this potential function, the agents can achieve and maintain the desired formation. As a consequence, the control law we have implemented for each agent i is the following:

$$\dot{x}_i(t) = f_i(x(t)) = -\sum_{j \in \mathcal{N}_i} \left(\|x_i - x_j\|^2 - d_{ij}^2\right)(x_i - x_j) \tag{2.3}$$

- Undirected connected graph - Collision avoidance

Conclusions