MODELING AND CONTROL OF CPSs - Project activity report (Part I)

Lorenzo AGHILAR (334086), *Carlo MIGLIACCIO* (332937), *Federico PRETINI* (329152)

Task #1: IST algorithm for solving LASSO

Brief theoretical introduction

When we have to solve an optimization problem, finding a *simple approximate solution* is surely better than an exact but more complex one. In our case **simplicity** corresponds to **sparsity**. A solution is said to be **sparse** if it has **few non-zero elements**. A *sparse optimization problem* can be formultated in the form $\min_{x \in X} \ f(x)$ s.t. $\|x\|_0 \leq h$, where x is the optimization variable and $f(x)$ the objective function. The main disadvantage of this formulation is the ℓ_0-norm which, being *non-convex and discontinuous*, makes the problem itself **hard** to be solved. Here the trick is the use of the ℓ_1-norm as an approximation, which has better properties, first of all the **convexity**.
In many real-world applications we need to find a **sparse solution** to a large underdetermined systems of linear equations. The presence of **measurement noise** in collecting data justify us in using a least-squares approach while the requirement on sparsity is fulfilled by introducing an ℓ_1-based penalty. This raises the problem of **LASSO** (*Least Absolute Shrinkage and Selection Operator*) defined as

$$x^* = \arg\min_{x \in \mathbb{R}^n} \frac{1}{2}\|y - Ax\|_2^2 + \lambda\|x\|_1, \ \lambda > 0 \tag{1}$$

where λ is an hyperparameter to tune. There are many algorithms to solve the LASSO problem. However, the objective of this task is to analyze the **Iterative Shrinkage/Thresholding Algorithm** also called **ISTA**.

Iterative Shrinkage/Thresholding Algorithm

The algorithm on which we focus our attention is derived from the alternating minimization of a *surrogate funtional* $\mathcal{R}(x, b)$ which is obtained adding some terms to the one in (1).

Algorithm 1 Iterative Shrinkage and Thresholding Algorithm (ISTA)

1. **Initialization** $x_0 \in \mathbb{R}^n$, e.g. $x_0 = 0$
2. For each $k = 0, ..., T_{max}$

$$x(k+1) = \mathbb{S}_{\lambda\tau}[x(k) + \tau A^T(y - Ax(k))], \quad \mathbb{S}_{\lambda\tau} = \begin{cases} x_i - \lambda\tau & \text{if } x_i > \lambda\tau \\ x_i + \lambda\tau & \text{if } x_i < \text{-}\lambda\tau \\ 0 & \text{if } |x_i| \leq \alpha \end{cases}$$

3. **Stop condition**: $\|x(T+1) - x(T)\| < \delta, \ \delta = 10^{-12}$

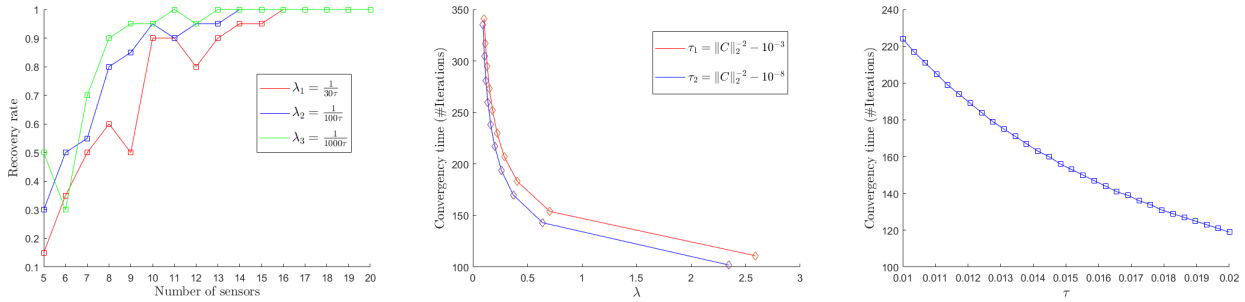
$\mathbb{S}_{\lambda\tau}$ is the **Shrinkage/Thresholding operator**. The parameter λ has a crucial role, in particular for $\lambda = 0$ we just get the LS estimate of the full model, while for *very large* λ the LASSO estimates are exactly zero. The **Algorithm (1)** has the following properties: (i) it is a **descent algorithm**, (ii) it converges to the minimum of LASSO; furthermore, its iterative nature makes it suitable for both **dynamic** and **distributed** settings.

Analysis and Results

In order to perform the analysis, data have been randomly generated: the entries of C are distributed according $\mathcal{N}(0, 1)$; the entries of the (sparse) vector x are assumed to be uniformly distributed; finally the solution is imposed to be 2-sparse. The **measurement phase** is distributed in the sense that the measurements have to be interpreted as taken by q sensors, each one of them taking one measurement.

First aspect: Support recovery rate

The **support** of a solution is the set of indexes at which the non-zero elements are placed. The analysis is performed on what is the **effect of increasing the number of sensors** q in the capability of the algorithm in recover correctly the support of the solution. To this aim, for each value of q from 5 to 20, 20 experiments have been executed.



(a) Support recovery rate (b) Convergency time (I) (c) Convergency time (II)

Figure 1: Results for the analysis of ISTA

It can be observed Figure (1a) that increasing the number of sensors the support recovery rate improves reaching also 100% for $13 \leq q \leq 6$. Moreover the experiments have been repeated for three different and decreasing value of λ.

Second aspect: Convergence time

The objective here is to discuss how the hyperparameters τ and λ can affect the performances of ISTA in term of **convergency time**, which - for our purposes - is the **number of iterations** needed to meet the stop condition. Two different types of experiments have been carried out: the first aimed to explain what happen when we change the coefficient λ of the regularization and keeping τ constant, the second in order to change τ, keeping $\tau\lambda$ constant, and see the related effects.

$\boldsymbol{\lambda}$	0.0800	0.0896	0.1017	0.1177	0.1396	0.1715	0.2223	0.3160	0.5457	2.0011
Iterations	621	590	560	529	456	404	367	328	287	236
Iterations	647	617	586	518	477	423	386	347	306	253

Table 1: Convergency time in function of λ

How can be noted from the Figure (1b), when we use an higher λ the convergency time is smaller. This is due to the fact that a larger λ drives faster the elements of the solution of the problem (1) close to zero, in this way the stop criterion is met sooner. However, the graph in (1a) shows that the performance in term of support recovery rate is worst on average. More detailed information can be found in Table (1).
On the other side, we have a similar situation when τ (step-size) is the parameter being changed: an higher τ leads to a faster convergence, this is related to the fact that in this case the distance between solution $x(T+1)$ and $x(T)$ for a generic instant T is larger. Figure (1b) shows that keeping λ constant, if we choose $\tau_1 < \tau_2$ the convergency time is on average higher. Finally the Figure (1c) shows the evolution of the numeber of iterations keeping the product $\tau\lambda$ constant.

Task #4: Dynamic SSE

Introduction

In the previous task we performed SSE on a static system, now we leave out the static hypothesis and move on to the dynamic one. Recalling the fact that a CPS with some sensors under attack can be described by means of system (1), in the dynamic case the matrix A is no longer the identity matrix but will become more complex.

Algorithms

In order to solve this problem we could think of using least square or Gradient Descent (**GD**) algorithms but in the former case it might be computationally complex to invert the C-matrix, while in the latter case the algorithm would be too slow to be applied to a dynamic case. In order to speed up the GD we can run

a single gradient descent step at each k instant, this algorithm has been called Online Gradient Descent or **OGD**

$$\hat{x}(k+1) = A\hat{x}(k) - \tau AC^T[C\hat{x}(k) - y(k)]$$

Now we can add the attack on the formulation and obtained the so called **augmented observability matrix** O_t. From the theory we know that if the matrix A as an eigenvalues equal to 1 the dynamic of the CPS also with constant attack is not observable. But thanks to the information about the sparsity of the attack we can develop a SPARSE OBSERVER in order to be able to solve the following problem

min....
after a sufficient small number of step T.
The algorithm of the sparse observer is the following:

Results

Task #5: Distributed Secure State Estimation of CPSs

Brief theoretical introduction

The framework in which CPSs are located is intrinsically **distributed**, for this reason we move in the direction of *fusion center removal* and the use of a distributed approach also for computation; this is done by using **distributed algorithms** which requires the devices of the CPS modeled as a multi-agent systems to cooperate, in the sense that they have to exchange some information through a communication network which is modeled by a digraph. Important results on **Consensus algorithm** guide us in the choice of the type of information to be shared: the local estimate $x^{(i)}(k)$ of the state. Under certain condition by iterating the computation of a **local quantity**, a global estimation x^*, equal for all the agents (the consensus), can be reached. The ideas on which Consensus algorithm is based can be exploited for the minimization of composite convex functionals like the Least-Squares and the LASSO one, which is used for our purposes. This brings us to a **distributed context** in the sense that if each node has got its own part of the composite convex functional, this is suitable to be used to compute and update local estimate also using the information sent by the nodes in its neighbourhood. This is the idea on which algorithms like the Distributed Gradient Descent (for the distributed solution of LS) and Distributed ISTA (for distributed solution of LASSO) are formulated.
The **objective of this task** is instead to solve the *RSS-Fingerprinting based target localization problem* in a distributed way using the DIST algorithm.

Algorithm

The algorithm, whose steps are summarized in the following, aims to find a solution to the problem (1) in a distributed way. Each agent has its own y_i and A_i which is the i-th row of the matrix A.

Algorithm 2 Distributed Iterative Shrinkage/Thresholding algorithm (**DISTA**)

1. **Initialization**: $x^{(i)}(0) \in \mathbb{R}^n$ (eg. $x^{(i)}(0) = 0$)
2. For $k = 1, ..., T_{max}$
3. For each agent $i = 1, ..., q \quad x^{(i)}(k+1) = \mathbb{S}_{\lambda\tau}\left[\sum_{i=1}^{q} Q_{i,j} x^{(j)}(k) + \tau A_i^T (y_i - A_i x^{(i)}(k))\right]$

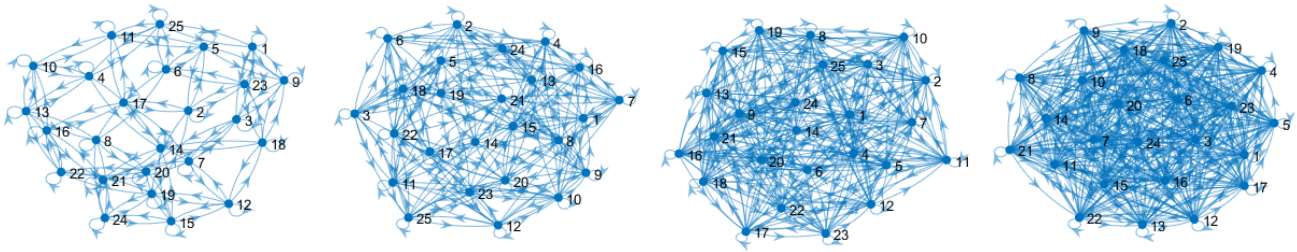
$\mathbb{S}_{\lambda\tau}$ is the **Shrinkage/Thresholding operator** used also in the previous tasks, while the matrix Q is the **stochastic matrix** associated to the directed graph through which the interconnection among the agents is modeled. The conditions under which such agents can reach the consensus can be verified by analyzing the eigenvalues of such a matrix according to the **Perron-Frobenius Theorem**. Moreover if the matrix Q has the properties required by such a theorem and it is *doubly stochastic* it reaches the average consensus, which is an even stronger result.
About the **cell grid discretization** and the structure of the solution, the setting is the same than the other tasks (both the state to estimate and the attacks are sparse).

Analysis and Results

In the following four different graph topologies with different connectivity properties have been proposed. For each one, the **spectral properties** and **the convergence** is analysed. In particular is useful to recall that:

1. A matrix is said to be (row) **stochastic** if the sum of the element for each row is equal to one; **doubly stochastic** if even the sum of the entries on the columns is equal to one;
2. The agents which communicate according to a certain Q matrix **reach the consensus** if $\lambda_1 = \lambda_{PF} = 1 > |\lambda_2| \geq ... \geq |\lambda_q$
3. The convergence of the consensus algorithm is determined by the **essential spectral radius** which is the **second largest eigenvalue in magnitude**.

The four matrices associated with the proposed topologies in Figure (2) have eigenvalues which satisfy the Perron-Frobenius Theorem since all are in magnitude less than $\lambda_{PF} = 1$, then they reach consensus. Moreover, since the matrices are even **doubly stochastic** the system reaches the *average consensus*. About



(a) Q_4 (b) Q_8 (c) Q_{12} (d) Q_{18}

Figure 2: Topologies on which the analysis is conducted

TOPOLOGY (Q)	**EIGENVALUES** $\lvert\lambda_i\rvert$	`esr(Q)`	**CONVERGENCY TIME**
Q_4	0.0450 ... 0.7785 1.0000	0.7785	10278
Q_8	0.0058 ... 0.4964 1.0000	0.4964	10498
Q_{12}	0.0053 ... 0.3701 1.0000	0.3701	10401
Q_{18}	0.0096 ... 0.2196 1.0000	0.2196	10637

Table 2: Main results for the analysis of the four topologies

the **convergency** we can state that a smaller value for the essential spectral radius is associated with an higher number of needed iterations to converge. This information is summarized in the Table (2)

Furthermore, The algorithm has a **good accuracy** since the estimate for both the support for the target position (progressive number of the cells) and sensors under attack are correct, the Figure (3) shows this result. Another consideration can be done on the entries of the stochastic matrices. In particular, trying to change the magnitude of the entries Q_{ij} there are not big differences on the convergency time. However in some situtions can be useful taking into account the weight of the edges. For example, let us suppose that on a link there is an attack in the sense that a certain sensor spread the measurement of the state corrupted by a quantity a, in that situation can be crucial - for the Secure State Estimation - the reduction of the quantity Q_{ij} associated with the link under attack.

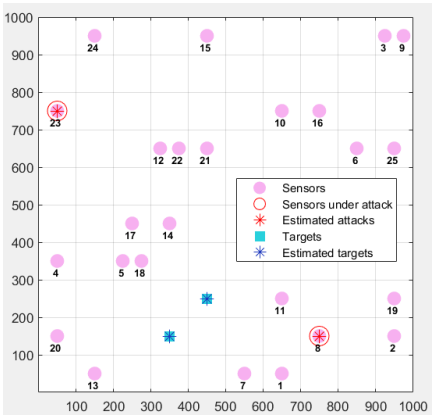


Figure 3: Sensors, Target, estimates