Modeling and control of cyber-physical systems
Project I

Merangolo Maria Francesca (s318905), Pisani Fabrizio (s305391),
Zampolini Matilde (s316182), Zulian Matteo (s310384)

a.y. 2022-2023

1 Task 1

The first task asks to implement the Iterative Shrinkage Thresholding (IST) algorithm to solve the LASSO problem and to estimate a sparse signal with noisy measurements.

To do that we set the hyperparameters that control the sparsity of the signal and initialize, with random values, the variables of the system such as the matrix `C` and the k-sparse vector `x_true`, which is the vector to be estimated, then add small noise to the measurement vector `y = C * x_true + nu`.

The goal is to estimate `x_true` knowing `y` and `C`, the IST starts from `x = [0,...,0]` and modifies its components at each step of the algorithm, when there are not significative modifications it means that the value has converged and the algorithm stops.

To see the effectiveness of IST, we run it 100 times with different values and save how many times the support of the estimate is equal to the support of `x_true`. With the initial values of the parameters it works correctly 60% of the times

Increasing the number of sensors q improves the performance of IST, as it will increase the number of measurements and thus provide more information about the underlying sparse signal.

However, there is no guarantee that increasing q will lead to 100% of success in the support recovery, especially if the signal is highly sparse or if the noise level is high.

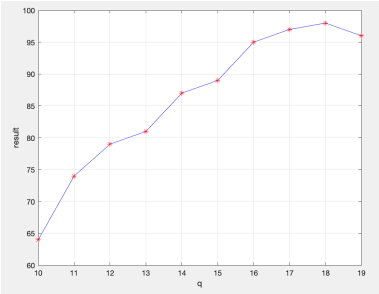

Figure 1: Percentage of success in support recovery increasing q

The plot shows that the minimum, mean and maximum number of iterations required to estimate the signal decreases as q increases:

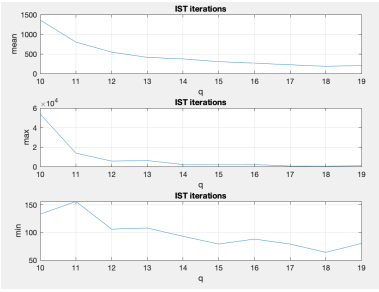

Figure 2: Mean - Max - Min number of iterations

The parameter τ is the learning rate and it is related to the convergence time. If we decrease τ the argument of the shrinkage/thresholding operator changes: $x_t + \tau C^T(y - Cx_t)$ making it less efficient, increasing the convergence time.

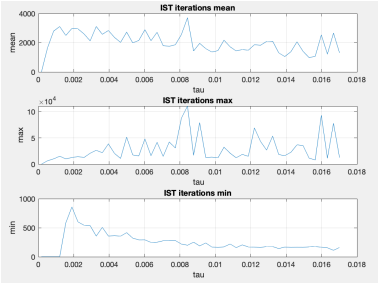

Figure 3: This plots must be read from right to left

When λ increases, the LASSO solver will consider more important to minimize the L1 norm $||x||_1$ rather than $||Cx - y||_2^2$, which contains the data of the problem.

This behavior will drive the the algorithm to choose, as optimal solution, a vector made of all zeros, in the worst cases the solver does not even complete one iteration because the initial condition is `x = [0,...,0]`, this is clearly a problem because the accuracy drops to zero.

IST is a powerful tool, but needs to work with the right hyperparameters to be accountable.

In conclusion, the choice of lambda should not be too high, but in the order of $10^{-1} - 10^{-2}$, allowing the algorithm to choose the right balance between sparsity of the solution and regard to the data of the problem.

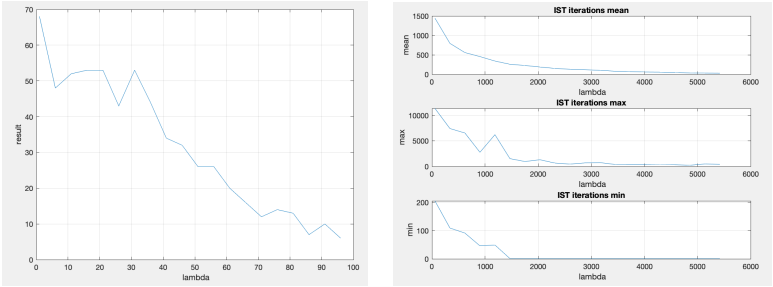

Figure 4: Increasing λ

2 Task 2

In Task 2 it has been used the IST algorithm to estimate the non-sparse x under sparse sensor attack.

The IST algorithm of this task is similar to the previous one. The most important difference is that `x_true` in this task is non-sparse.

The algorithm is run in four different scenarios. The variables `aware` and `nu_flag` are used to differentiate the cases of aware and unaware attacks with and without noise.

An unaware attack is an attack in which the attacker has no knowledge: for this reason, the type of attack is modeled with a random uniformly distributed value in the range $[-2, -1] \cup [1, 2]$. An aware attack is an attack where the attacker has information about the system: because of this, the attack a is model as $\frac{1}{2}y$. When `nu_flag=1`, a small noise is added to the measurement vector.

For each scenario, the algorithm has been run 100 times.

At the end of every run, there is a computation of the `j-th` component of the vector `result` and of `result_m` (estimation accuracy of the run). At the end of every scenarios (100 runs), there is a computation of `result_a` (the l_1-norm of the vector result, i.e. the amount of times the support of the attacks' vector is correctly estimated) and `result_mean` (i.e. the mean of the estimation accuracy of the scenario).

The number of correct estimations is in the vector `result_a`. The four scenarios produce different results. The best scenario (in terms of estimation of the support of a) is the one with unaware attack and without noise: this is due to the fact that the a is randomly generated. The worst scenario is the one with aware attack and not zero noise: this is due to the fact that the measurement y is strongly affected by the noise and an attack which is proportional to y.

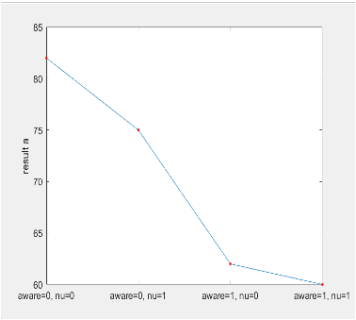

Figure 5: Percentage of success in support recovery

The estimation accuracy is in the vector `result_mean`. Running several times the algorithm, it turns out that the four scenarios produce almost similar results (about 0.1). We can consider it a good result.

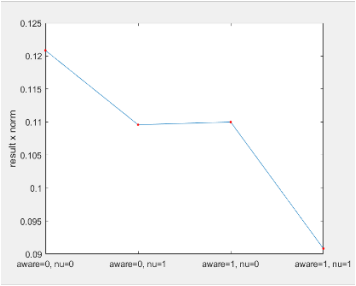

Figure 6: Estimation accuracy

Overall, the results show that the IST algorithm can estimate a non-sparse x under sparse attack (in four different cases).

3 Task 3

The aim is to detect the presence of targets and estimate their locations using the RSS measurements collected by sensors located at different positions.

The first part of the code focuses on target detection without any attacks. Initially, we normalize a dictionary matrix D, which represents the distances between the sensors. Then, we apply the IST algorithm to solve the LASSO problem with a regularization parameter lambda. The resulting sparse vector `x_t` indicates which cells contain targets, with non-zero entries representing target presence.

However, the algorithm produces false positives because there is little data available. Specifically, in the absence of attacks, the code detects 3 targets located at positions 2, 5, and 7. However, the value in cell 7 is significantly greater in absolute value than the others, suggesting that the target is present in that cell. To improve the accuracy of target position estimation, we have introduced a tolerance, defined as `max(x_t)/2 - 1`. This tolerance helps identify false positives by setting the lowest values of it to zero. As a result, only the target in position 7 is detected.

Subsequently, a simulated attack on a single sensor is introduced by adding a perturbation component, denoted as `a(h)`, to the measurement `yi` of the h-th cell. This perturbation is equal to $\frac{1}{5}y_i$ for each iteration. Then we normalize the matrix G, which is obtained by appending the identity matrix to D. Once again, the IST algorithm for solving LASSO is applied to the updated measurement data in order to estimate a new sparse signal, `x_t_Attack`. This signal incorporates both the target positions and the simulated noise, `a_est`.

Similar to the first part of the code, we have set a tolerance value to prevent false positives in `x_t_Attack`.

In this case, the algorithm encounters false positives not only in identifying the target positions but also in determining which sensors are being attacked (`a_est`). To solve this issue, we have introduced a new tolerance equal to 2.

The code tracks the number of targets detected under attack in each iteration by comparing the indices of the non-zero entries in `x_attack` with those obtained in the previous section. This evaluation allows us to assess the algorithm's robustness, which in turn can be leveraged to optimize the fingerprinting strategy and design more resilient sensing systems.

Analyzing the results, it can be seen that in each iteration a target is consistently detected at position 7, just like in the first part of the code without an attack. Furthermore, in most cases, the estimated values of the attack are close to the actual values, indicating that the LASSO algorithm is capable of discretely estimating the attack. It successfully identifies the attacked sensors, except for sensors 3 and 4. This discrepancy is due to the attack corrupting the measurements and introducing additional non-zero entries in the sparse vector.

4 Task 4

The goal of this task is to track the unknown state $x(k)$ that is moving in a square room of 100 cells, given the matrices A, D and the vector of sensors' measurements $y(k)$.

We begin by defining a binary vector for the true state with a random support. It defines the initial locations of the targets: if $x_i(k) = 1$ there is a target in cell i.

In this case, the state x is not a unique vector but a set of vectors that describes a dynamical system.

The main feature of this online estimation algorithm is that at each time step we also update the current measurement $y(k) = Dx(k)$.

We consider a dynamic system with time-varying attacks. However, we assume that the support of $a(k)$ is constant over time.

We define the matrix G and the vector $\hat{z}(k)$ as follows:
$\hat{y}(k) = D\hat{x}(k) + \hat{a}(k) = G\hat{z}(k)$, where $G = \begin{bmatrix} D & I \end{bmatrix}$ and $\hat{z}(k) = \begin{bmatrix} \hat{x}(k) & \hat{a}(k) \end{bmatrix}^T$

We have to do a normalization of the matrix G because columns of D are more powerful and, otherwise, all the elements related to the second part would be put to zero.

The important difference of the sparse observer algorithm from the IST one is in the fact that at time k we perform the computation of an intermediate step $\hat{z}(k + \frac{1}{2})$ by the shrinkage/thresholding operator. This intermediate state is a refinement of the state $\hat{x}(k)$ computed before time k (prediction). Then, we separate x and a part:

- For x part we proceed with prediction exploiting the knowledge of matrix A.
- For a part we have no other information except the sparsity. We take the vector a as it is for the next step. However, we consider only the components with magnitude below 2. In a less simplified context, we could also apply the shrinkage/thresholding operator.

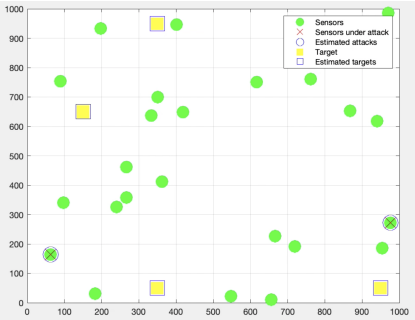

Figure 7: State tracking

We can see that the first estimates are not very good, but we can converge quite fast to the correct state.

The tracking of the targets is quite good for both the cases of aware and unaware attacks. However, in terms of detecting attacks, adding a non-constant but proportional quantity, we find a relevant number of false positives (we see many other attacks).

In this case, the attacker knows the matrix D because it corrupts the sensor measurement y_i of a proportional quantity. Therefore, in the measurement we get something that is in the image of D, so

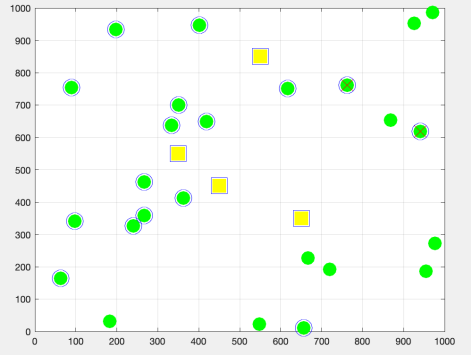

Figure 8: Aware attacks

we cannot detect the true attacks, while in practice some perturbation of x was done. This can also be due to the fact that now the attacks are time-varying.

We are not very good in the attack detection but the final goal which is the secure estimation of the targets in the presence of attacks is quite correct.

5 Task 5

The task is an implementation of the distributed IST (DIST) algorithm for secure estimation in a network with sparse sensor attacks. We start by checking the consensus property of four given stochastic matrices Q, which represent different network topologies, then we generate random data for the problem, x, C, the (unaware) attack vector a, we initialize the parameters (lambda, tau, ...) and other variables like z, which is the union $\begin{bmatrix} x & a \end{bmatrix} \in \mathbb{R}^{n+q}$.

We perform the DIST algorithm which works like this: starting from zero, at each step, each sensor i computes a new estimation $z^{(i)}(k+1)$, using the shrinkage operator, based on the $z(k)$ that the neighbor sensors shared to the sensor i and its local information (y_i and C_i). It goes on until there is no significant change between one step and the previous.

1-2) After the algorithm converges we extract from z the vector x_{tilde}, calculated as the mean among the state vectors that each sensor has estimated, and we analyze the results obtained: every sensor converged to the same values so they reached the consensus and their result is quite accurate as we can see by the norm $\|x_{tilde} - x\|^2$ which is very low (10^{-2}).

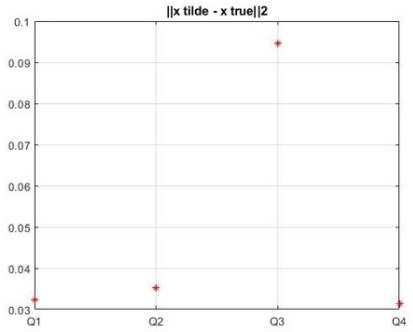

Figure 9: Estimation accuracy

3) For what concerns the estimation of the attack vector we can look at the last q columns of z and compare the results with the real a vector and we can see that the support is estimated correctly for every sensor. The topology, so Q, doesn't change the accuracy of this prediction.

4) By analyzing the convergence time of the four topologies, we notice that it increases as the essential spectral radius of Q increases.

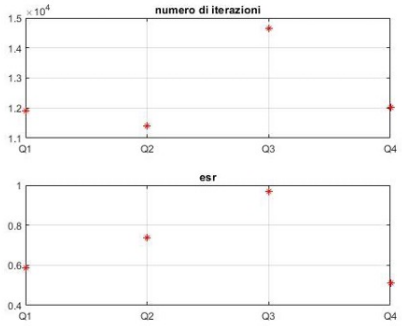

Figure 10: Relationship between esr and convergence time