Network Dynamics and Learning: Homework 3

Fabio Tatti
Politecnico di Torino
s282383@studenti.polito.it

1. Exercise 1

We simulate a SIR epidemic on a symmetric *k*-regular undirected graph with number of nodes $n = 500$, each one with number of closest neighbours equal to $k = 4$.

1.1. Problem 1

Let the parameters for the epidemic to be $\beta = 0.3$, representing the probability for a infected individual/node to infect a susceptible neighbour if infected. The other parameter is $\rho = 0.7$, representing the probability that an infected individual/node recovers from the disease. Both the previously mentioned parameters refer for a time step.The simulation will be of 15 weeks, given an initial configuration of 10 infected nodes, and all the remaining susceptible. In order to achieve this goal, we first implement the function *build_symmetric_graph*, that given the number of nodes and the degree of link of each one of them will return a symmetric *k*-regular undirected graph object. An example is shown in Figure 1. Then we create the custom function *simulate_infection* that, given the Graph on which simulate and the initial number of infected nodes will simulate the spread of the simulation for a number of weeks (steps) equal to 15. The function will return a list of list, where each nested list is the state at a given week with entries:

$$[week, I\_week, B, S, I, R]$$

where *I_week* are the new infected in the given week, *B* are the number of links between a susceptible node and a infected one, *S* the total number of susceptible nodes at the given week, *I* the total number of infected nodes at the given week and *R* the recovered. Given that function we can easily simulate a multiple number of times $N = 100$ the 15 weeks epidemic, in order to print:

- The average number of new infected individuals for each week (for us *I_week*) as shown in Figure 2 .
- The average total number of susceptible (*S*), infected (*I*) and recovered (*R*) individuals at each week as shown in Figure 3.

We did not make usage of matrices for simulating these epidemics, preferring to use a simple dictionary of lists to keep track of all the different states of the model. Moreover, choosing of the first 10 infected nodes was done with a random sample from all the nodes of the graph.

1.2. Problem 2

In this part we will focus on the generation of a random graph, done by exploiting the *preferential attachment model*. We aim to create a random graph with average de-

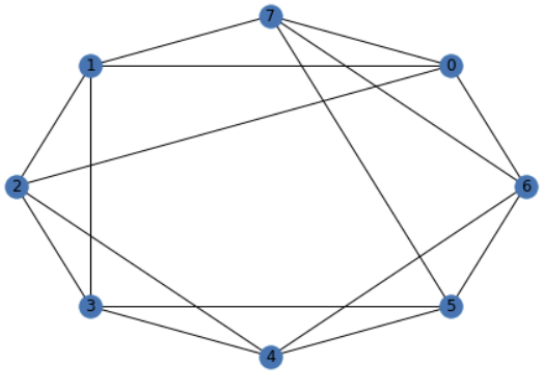

Figure 1: Example of graph returned by the custom function *build_symmetric_graph* given $num\_nodes = 8$ and $k = 4$.

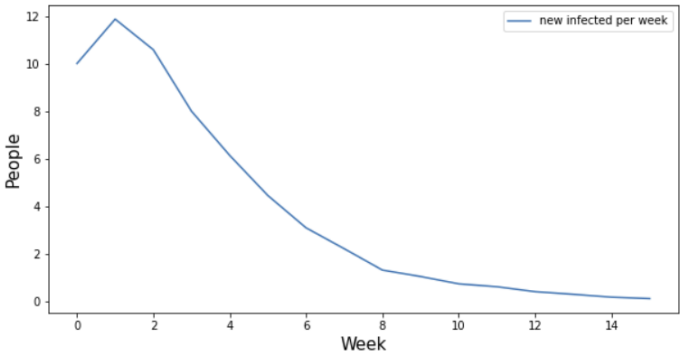

Figure 2: Average number of newly infected individuals each week for Problem 1.1.

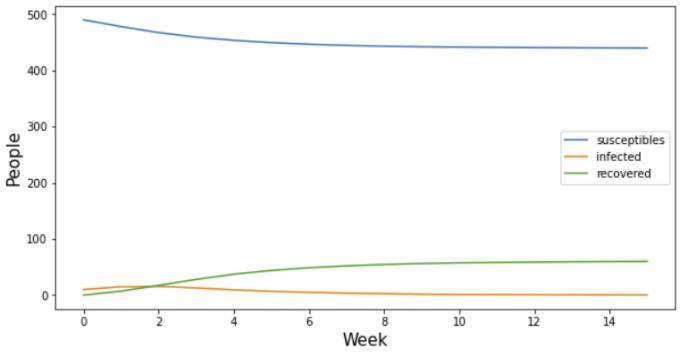

Figure 3: The average total number of susceptible (S), infected (I) and recovered (R) individuals at each week for Problem 1.1.

gree close to $k = 4$ (our implementation is capable to generalise also to other values). This is done with an iterative method, that adds a new node at each iteration to an initially complete graph with $k + 1$ nodes. The function able to create this kind of random graph is *gen_random_graph*, that takes as input the number of nodes and the wanted average degree *k*.To obtain the right degree of the final graph, each time we add a new node we set the probability to connect it to an already present node to be equal to be proportional to the degree of the already present node.

$$\mathbb{P}_i = \frac{w_i(t-1)}{\sum_{j \in \mathbf{v}_{t-1}} w_j(t-1)} \tag{1}$$

where $i \in V_j$ is any already present node, and $\mathbb{P}_i$ is the probability to connect to it. In our implementation we did not use any kind of matrix, it would be too much expensive: instead we make a wise use of lists. Our version is capable to deal with both odd and even values for k, of course integer, as shown in the notebook.

2. Exercise 2

Now we want to simulate a pandemic without vaccination by exploiting all the functions previously created during the homework. First we generate a random graph by using *gen_random_graph* with $num\_nodes = 500$ and $k = 6$. Then, we simulate for $N = 100$ times 15 weeks of epidemic by using the function *simulate_infection* with 10 initial infected nodes. The results are processed in order to print:

- The average number of new infected individuals for each week (for us *I_week*) as shown in Figure 4
- The average total number of susceptible (*S*), infected (*I*) and recovered (*R*) individuals at each week as shown in Figure 5.

We can appreciate by comparing those new graphs with the ones of the previous exercise how the topology of the graph affect drastically the spread of the disease. This comes naturally when we think that, as result of the *preferential attachment model*, some nodes will have an high degree and the more high it is, the more high will statistically get while adding new nodes to the graph. High degree means also high number of infections in our model, that explains why, respect to a graph where all nodes have the same low degree, in this case we register an higher peak of infections. Moreover we also can notice how the peak is delayed in time respect to Figure 2, and how the population is more involved in the pandemic, seen how in Figure 5 the curve of recovered people (meaning that has contracted the disease) goes in the range of 400 people.

3. Exercise 3

We do the same as before, but taking action by using the vaccination to slow down the spread of the virus in the population. A given percentage of the population is vaccinated each week: those people are randomly selected among the population that has not received the vaccine yet. Once that a person is vaccinated he/she can not infect or get infected: this means that in the specific case that a in-

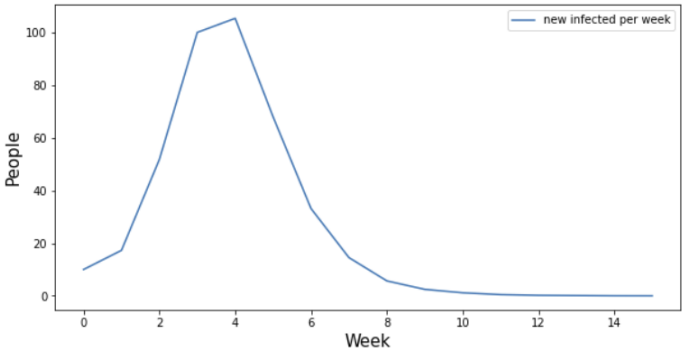

Figure 4: Average number of newly infected individuals each week for Problem 2.

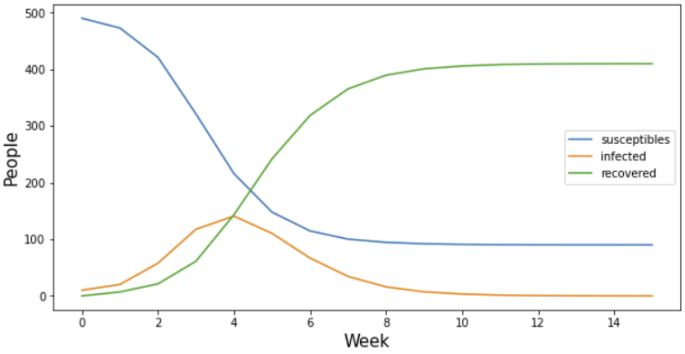

Figure 5: The average total number of susceptible (S), infected (I) and recovered (R) individuals at each week for Problem 2.

fected person gets the vaccine we assume that recovers from the disease immediately. We also vaccine recovered people. In order to accomplish that task we create a new version of the *simulate_infection* function that we call *simulate_infection_vaccine*: this new code will simulate a pandemic where, at the beginning of each week (step), a given percentage of the population is vaccinated. So we consider a new state and also return a new resulting list of list where each nested list is:

$$[week, I\_week, B, S, I, R, V, V\_week]$$

where the new entries are *V* that represent the total number of vaccinated people and *V_week* is the number of people vaccinated in the given week. We simulate $N = 100$ times the spread of the epidemic with the vaccine: each iteration will have a 15 weeks (steps) range, and then we store and process the data in order to average and get the plots of:

- The average number of newly infected and newly vaccinated individuals each week, as shown in Figure 6
- The average total number of susceptible, infected, recovered and vaccinated individuals at each week, as shown in Figure 7.

We can notice how the average number of new weekly infected drops earlier respect to the simulation where vaccination was not taken into account. At the same time we can also notice from the Figure 7 that the number of infected people is lower respect to the previous simulation, due to the action of the vaccine that decrease the fastness of spreading of the disease.

4. Exercise 4

All the previously used functions will be used in this section to estimate the social structure of the Swedish population and the disease-spread parameters during the H1N1. The reduced population on which we will simulate the

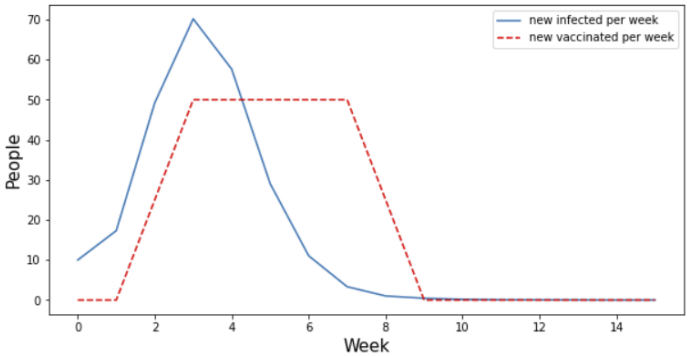

Figure 6: Average number of newly infected and vaccinated individuals each week for Problem 3.

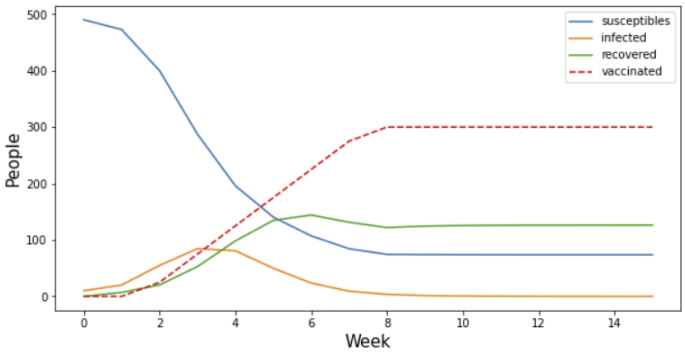

Figure 7: The average total number of susceptible (S), infected (I), recovered (R) and vaccinated (V) individuals at each week for Problem 3.

disease spread will have 934 nodes, a scale down factor of 10^4 respect to real size of the Swedish population. We want, given the number of vaccinated over all the 15 weeks, the population graph and an initial number of infected to find the values for parameters ρ and β. This will be done by predicting the number of new infected over all the weeks and comparing to the real value: this will allow to choose iteratively the best parameters. We choose to decrease gradually the values for $\Delta\rho$ and $\Delta\beta$ of a factor equal $number\_iteration/2$ to ensure the convergence; that was not done for Δk since k is an integer number. The metric used to choose the best parameters at each iteration is the Root Mean Squared Error (RMSE), computed each time on the prediction with respect to the real value for the new weekly infected. Each configuration of the parameters $[k - \rho - \beta]$ is simulated $N\_regression = 10$ times in order to be representative. We also decided to set a threshold number of maximum iterations that the simulation can take, to prevent it to run for infinity since the convergence to $RMSE = 0$ seem to be hard to obtain. Initially we set the parameters to be $k = 10$, $\Delta k = 1$, $\beta = 0.3$, $\Delta\beta = 0.1$, $\rho = 0.6$ and $\Delta\rho = 0.1$. The **best results** we get are:

- $k = 14$
- $\beta = 0.10813445131170563$
- $\rho = 0.4912506796819136$
- $RMSE = 3.2344821842143445$

Then what we did was to validate the result: we did it by simulating for $N = 100$ times the pandemic with the founded best parameters and then averaging the results in order to compute the RMSE. Each time we create a new random graph on which we simulate the pandemic. The resulting graph is the one showed in Figure 8 with a resulting:

$$RMSE = 4.822413814678288$$

The results is not too much disappointing in terms of error, especially if compared with the result that was given initially, when searching for the best parameters. The shape of the graph, instead, is similar to the ground truth but with some differences , even if the peak of infected is smoother. We also have to consider that the results may change over different runs, since they are affected by the randomness of the entire system.

5. Exercise 5

We try to improve the results of the previous exercise by changing the generation of the graph: we generate a *small-world* random graph by using the custom function *get_small_world*. We start from a symmetric *k*-regular undirected graph; then what we do is to add l additional undirected links, where l is a binomial random variable with parameters $nk/2$ and $p \in [0, 1]$. The nodes involved by those new links are randomly chosen indipendently and uniformly. We also choosed to iterate for $N\_regression = 100$, in order to have a more consistent result when validating. The first thing we notice are, of course, the higher computing times: those are the direct result of using the *small-world* approach, since we are introducing a new dimension in the latent space of the optimization for the parameter p. Moreover, also the $N\_regression = 100$ influence, leading us to test each time 8100 different graphs: 3^4 possible configurations given the four parameters, each one tested $N\_regression = 100$ times. In the previous exercise we where testing just $3^3 * 10 = 270$ different graphs. This huge computational gap is also influencing converging times, since with more parameters we will need more iterations to converge to a minimum: this is the main reason why we do not surprise when looking at the new scores in the validation given the best performing parameters:

$$RMSE = 6.688657843693307$$

given the search parameters:

- $k = 10$
- $\beta = 0.47061654897103833$
- $\rho = 0.9009205122811261$
- $p = 0.044826330961779876$
- $RMSE = 3.2344821842143445$

In order to compare those results with the previous we kept the same validation procedure as in the fourth exercise. Those results are of course disappointing: as shown in Figure 9, even in the curve we can notice a worse prediction of the number of new infected. Still, we can imagine that, if trained for a higher number of iterations, the model would probably perform better, since an additional parameter would increase the precision of the model, even if could appreciate it, due to our available power (40 iteration of the process took almost 24 hours!). Another thing that we noticed, also in the previous exercise, is the abundance of different configurations returning similar results, that leads us to think that the function we are performing the regression on may have an high number of local minima, also sustained by the small changes of the loss during the iteration process. We can conclude by saying that other variations could be done also on the loss, since we are not sure that RMSE is the best one for this given task: we did not try because results would not be numerically comparable respect to the previous exercise, just graphically in the plots. Even training with a new loss and validating with RMSE would probably have a bias, since the model would be optimizing on another loss rather the one we are referring to.

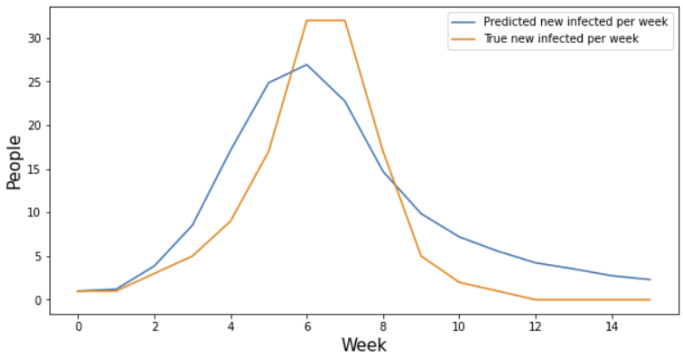

Figure 8: Average number of predicted and actually true newly infected for each week in Problem 4. RMSE = 4.822

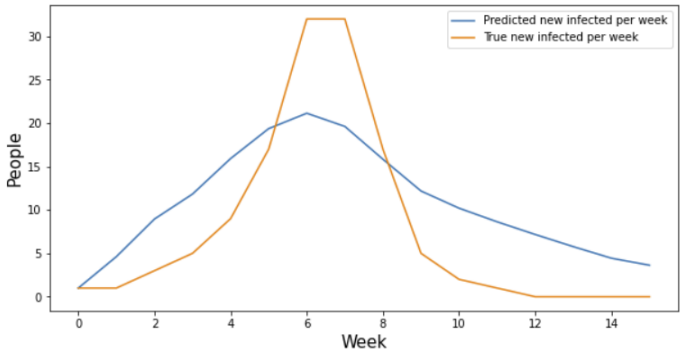

Figure 9: Average number of predicted and actually true newly infected for each week in Problem 4. RMSE = 6.689