NeuralPALS - Learning Exposure Functions and Infection Probabilities for Contagion Spread

by

Advaith Anand

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author . . . .
Department of Electrical Engineering and Computer Science
May 23, 2019

Certified by. . . . .
John Guttag
Professor
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

NeuralPALS - Learning Exposure Functions and Infection Probabilities for Contagion Spread

by

Advaith Anand

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Understanding how contagions spread is an important task, particularly when considering infectious diseases. An individual's likelihood of getting infected by a contagion is determined by a combination of inherent susceptibility and exposure to other individuals who may spread the disease. In a real world setting, an individual's infection status may be directly observable, but it is difficult to identify whether an individual is spreading the disease. As a result, the exact influence function by which disease is transmitted is difficult to understand as well. We present a neural network based method, NeuralPALS, to learn the spreader, exposure, and infection status of individuals in a network. Unlike previously developed methods, we do not assume an exposure function and instead devise methods to learn this function. Through experiments on synthetic data we illustrate our method's efficacy in determining both spreader and infection states. We also demonstrate NeuralPALS's ability to learn different exposure functions. In addition, we utilize a dataset of patients from a large urban hospital and demonstrate our preliminary results in determining the spread of Clostridioides difficile.

Thesis Supervisor: John Guttag
Title: Professor

Acknowledgments

First and foremost, I would like to thank my advisor, John Guttag, for giving me the opportunity to work with his group and for his feedback and advice. John's advising has led me to grow as both a researcher and a student through this year. Second, I'd like to thank Maggie Makar, who directly mentored me through the course of this project. Maggie guided me through the year and was always incredibly patient, kind, and helpful while. Without her help none of this would have been possible and I am incredibly grateful for it. Last but not least, I have to acknowledge my parents and sister - I can not possibly thank them enough for their constant support and encouragement.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Understanding how contagions spread remains an increasingly important task in the world around us. Contagions, while often associated with infectious disease, can also be understood as opinions, products, and more. The likelihood of a given individual contracting a contagion can be understood by analyzing both the individual's personal characteristics as well as the influence of potential contagion spreaders that an individual comes into contact with. For example, disease spread is contingent on both an individual's susceptibility and their contact. Political opinion is also derived from both an individual's own line of thinking and the opinions of their friends and contacts.

In medical settings, we frequently have access to a patient's individual characteristics, allowing us to determine their inherent susceptibility. Unfortunately, we may not always completely know the contagion states of an individual's neighbors, preventing us from possessing a comprehensive understanding of the individual's likelihood of contracting the contagion. This is particularly true in the presence of asymptomatic spreaders - individuals who may not show symptoms of a contagion but possess the ability to spread it. Focusing on the example of infectious diseases, it's vital for healthcare providers to understand which patients might be spreaders of a disease, in order to limit contact of other patients with these spreaders. Spreaders who do not exhibit symptoms, referred in this work as latent spreaders, pose a special problem for healthcare providers. While identifying latent spreaders is important, understanding how exposure is determined remains critical as well. For example, if exposure to just one spreader is enough to significantly raise the contagion risk of a patient, healthcare providers would act differently than if exposure to multiple spreaders is required for a patient to acquire a contagion. In this work we aim to not only find latent spreaders in a disease network, but also to understand the influence functions present in these networks.

1.2 Healthcare-Associated Infections and Clostridioides Difficile

Healthcare-associated infections (HAIs) are infections acquired by a patient in a healthcare setting, such as a hospital or clinic. HAIs are responsible for over 100,000 deaths per year in the United States, with over 1 in 31 hospital patients acquiring an HAI [3]. Clostridioides Difficile (*C. diff*) is a major HAI, a bacteria that infects the colon, leading to the deaths of over 15,000 Americans every year [2]. *C. diff* is spread through the fecal-oral route, with a common vehicle being contaminated surfaces in hospitals. Understanding latent spreaders of *C. diff* and its contagion mechanisms could lead to better healthcare outcomes, in which care providers could separate latent spreaders from potential victims.

1.3 Problem Definition and Contributions

In this work we aim to find latent spreaders and understand patterns of exposure in networks. While this work specifically focuses on an infectious disease setting, note that the work can apply to other examples, such as deciphering the spread of political opinion through social networks [6]. We work with a snapshot of a community - that is, observing the state of individuals at a given point in time, and develop a neural-network based model to predict the probability of individuals getting infected.

Inputs to our model include individual patient characteristics, and an adjacency network consisting of contact between patients. Through this, we develop the following contributions:

- **Predicting Individual Spreader States** - We utilize an individual's characteristics to predict their spreader state. Specifically, we learn a set of weights that map from individual characteristics to the probability of being a spreader.
- **Learning Exposure Functions** - We experiment with different possible exposure functions to understand contagion spread. Different contagions spread through different mechanisms - as a result, being able to modify the exposure function of a model is critical to working with different contagions. An example of a naive exposure function is simply averaging the spreader states of an individual's neighbors, and thresholding the exposure state based on this. For example, if an individual has 5 neighbors, 3 of which are spreaders, then their exposure state would be 0.6, which we could threshold to 1 - as a result, the individual is understood to be exposed.
- **Predicting Individual Infection States** - Through a combination of individual characteristics and the exposure derived above, we learn the probability of an individual contracting an infection. This approach utilizes both susceptibility and exposure to develop a model of disease transmission. As with the spreader states, we learn weights that correspond to a mapping from the characteristics and exposure to the probability of infection, which provides an understanding of the importance of exposure on infection probability.

1.4 Outline

In the following chapters of this thesis, we will cover the following work. To begin, in Chapter 2 we cover technical background material and detail a brief review of related literature. In Chapter 3 we discuss the synthetic data generation process and the NeuralPALS model. We then detail experiments run on both synthetic data and real patient data in Chapter 4, explaining. Finally, in Chapter 5, we re-examine the primary finding of this work and describe avenues for future experimentation.

Chapter 2

Background and Related Work

2.1 Technical Background

In this section, we present necessary background information on both the model we develop, NeuralPALS, and the data utilized.

2.1.1 Neural Networks and Permutation Invariance

Neural networks have been used in machine learning applications for many years because of their effectiveness as universal function approximators - a relatively simple neural network can represent a wide range of functions. Neural networks have been utilized throughout different fields, including numerous medical applications [1]. Neural networks have traditionally been developed to train on vectors and other forms of ordered data, with columns of input matrices representing different features. However, there are many learning tasks where input order is irrelevant. For example, if an input is a list of numbers and the goal is to find the sum, the order of numbers in the list does not modify the final result. We classify these sorts of learning problems as *permutation invariant.* A function, f, is permutation invariant for a given permutation α if $f(x_1, x_2, ..., x_n) = f(\alpha(x_1), \alpha(x_2), ..., \alpha(x_n))$

Recent work from Zaher et al. has established an architecture for the class of permutation invariant problems [7].

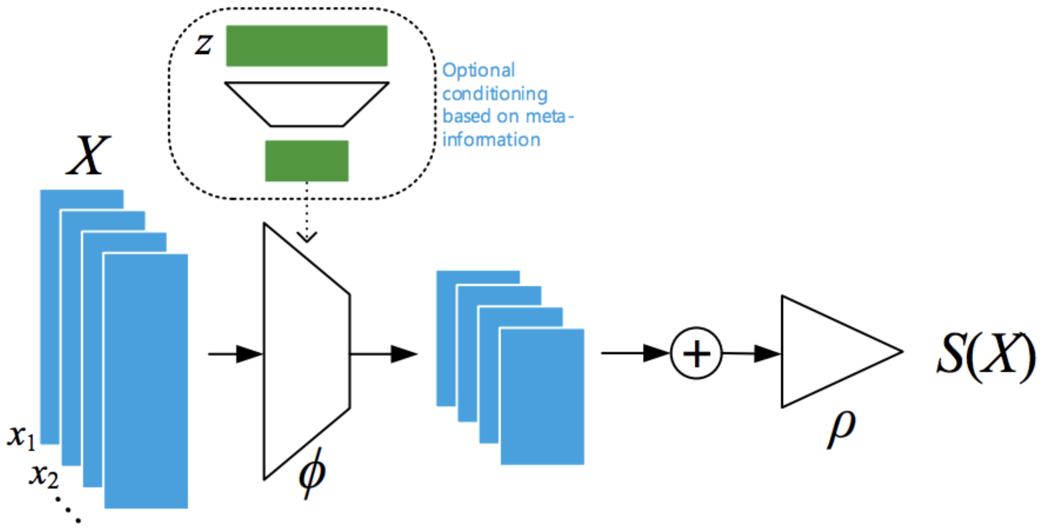

Figure 2-1: DeepSets Architecture [7]

The DeepSets architecture consists of converting inputs to latent representations, applying a permutation invariant function on the latent representations, and then applying a non-linear transformation. The architecture is illustrated in Figure 2-1.

Assume input X, consisting of m data points, $x_1, x_2, ..., x_m$. The ϕ layer transforms each input to a latent representation, $\phi(x_1), \phi(x_2), ..., \phi(x_m)$. The DeepSets authors prove that addition in the latent space is a permutation-invariant function. After summing the latent representations, the authors apply another network, ρ, leading to the final output, $S(X) = \rho(\sum_{i=1}^{m} \phi(x_i))$. The authors also introduce methods to condition the network on prior information but we do not use these in NeuralPALS. We employ the DeepSets architecture to learn exposure states, which we explain in detail in Chapter 3.

2.1.2 Synthetic Data

Synthetic data generation is commonly used when designing and evaluating predictive models. Synthetic data allows researchers to devise specific experiments and test various parameters without having to seek out new real-world datasets, a process that is often difficult and time-consuming. Synthetic data is particularly valuable for this work because of the nature of the problem - latent spreader states are, by definition, not observed. As a result, it is often difficult to verify the accuracy of a model in predicting latent spreaders in a real world dataset.

2.1.3 Hospital *C. diff* Dataset

To properly understand the success of our model in the real world we use a dataset from a large urban hospital, comprising over 70,000 total patients. As mentioned in Chapter 1, we specifically study *C. diff.* infection. We utilize the following selection criteria to develop a dataset for modeling. There are two sub-populations to select - the main cohort (to learn infection states) and the neighbors of these cohort members (to learn spreader states). We use similar data selection criteria to the PALS work described below. We only consider hospitalizations from May 2012 to May 2014.

Cohort

We assign an *infected* status to cohort members based on if they are diagnosed with *C. diff* infection after their fifth day in the hospital. This threshold, though somewhat arbitrary, is chosen to limit incidences of C. diff infection contracted outside of the hospital. Patients who are confirmed to have *C. diff* infection before the threshold are removed from our cohort. For all patients in the cohort we extract the infection state and individual characteristics. Individual characteristics comprise recorded information about patients, including medications, lab results, and more. All values are binarized.

Neighbors and Adjacency Network

C. diff infection is frequently spread through contaminated surfaces. In a hospital setting, vehicles for transport include rooms and nurses. *C. diff* may be transmitted from a patient to another patient sharing the same room. Similarly, a nurse, who visits multiple patients in a day, may transfer *C. diff* between patients. As a result, we construct two adjacency networks - a *room* network, with an edge between patients who shared time in the same room, and a *nurse* network, with an edge between patients if the same nurse administered medication to them on the same day. We exclude neighbors who are already in the main cohort, and then extract individual characteristics for the remaining patients.

2.2 Related Work

Contagion spread has been a popular avenue of research because of its importance in public health. Traditional epidemiological models of contagions include "susceptible, infected, removed" (SIR) and "susceptible, infected, susceptible" (SIS) models. These models are *compartmental* in nature, grouping individuals into categories - a simplistic approach that is useful for understanding population dynamics. In this work we strive to achieve a more granular understanding of contagion spread. The SIS and SIR models have been extended to to an individual-scale but these extensions have been unable to effectively learn latent spreaders [4].

2.2.1 PALS

Makar et al. devised a variational inference algorithm, PALS, to learn latent spreaders and infection states [8]. We build off the work in their paper in our research. The synthetic data generation process, described in Chapter 3, is based on the process in PALS. In addition, we utilize a similar real-world dataset.

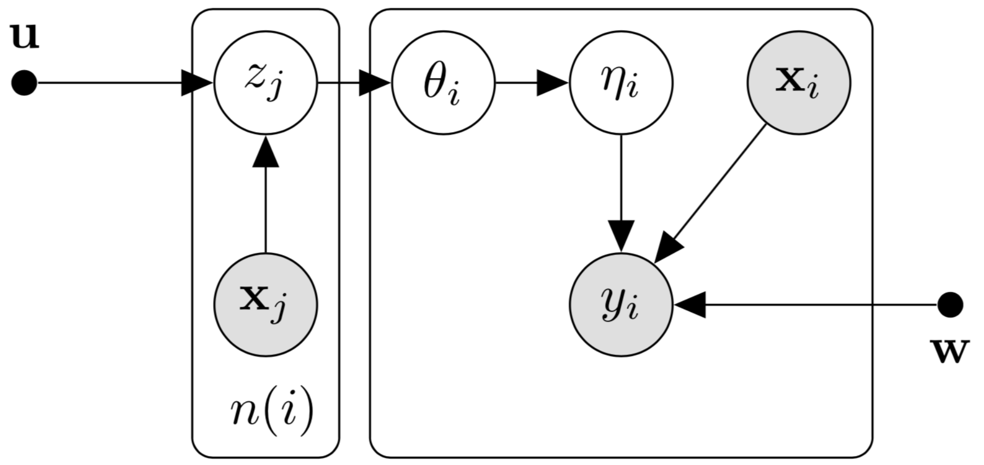

Figure 2-2: PALS Generative Model [8]

Variational inference essentially consists of finding the closest possible distribution to the "true" one by learning ideal parameters. PALS utilizes the generative model seen in Figure 2-2. In the model, z_j is a Bernoulli random variable that models whether a neighbor, j, is a latent spreader of the disease. x_j models the neighbor's characteristics and u is a set of weights. The probability of exposure, $\Theta_i|z_j$is a Beta variable, and the infection state of the individual modeled, i, is a Bernoulli variable.

We do not explicitly detail the PALS algorithm here, but the authors demonstrated its ability to effectively learn latent spreaders and infection states on both synthetic and real-world data. However, PALS assumes a simple exposure function - an individual is considered "exposed" if more than half of their neighbors are spreaders. In contrast, we want to learn different exposure functions dynamically - for example, if an individual needs to be in contact with just one spreader to be exposed, or if probability of exposure goes up non-linearly with each additional spreader contact. In this work we aim to perform at least comparably to PALS on learning latent spreader states while improving exposure calculations.

Chapter 3

Methods

3.1 Data

In this work we utilize both synthetic and real patient data. Synthetic data provides an easily modifiable process to investigate the performance of different models under varying parameters. The real-world data is used to assess the accuracy of models in a potential application.

3.1.1 Synthetic Data Generation

For this work we desired a few characteristics for the data generation method. To begin, the method must be transparent and easily modifiable. The method must be probabilistic and non-deterministic to mimic noise in real-world conditions. Finally, the method must control the probability of infection given both susceptibility and exposure. As a result, we based our data generation model on those described in the original PALS work, with slight modifications detailed below [8].

As mentioned previously, in order to predict individuals' infection statuses we desire to understand the impact of both susceptibility characteristics and of exposure to disease spreaders. We make the assumption that spreader status and susceptibility are independent. As a result, we must generate the following:

- Spreader and susceptibility characteristics

- Spreader, susceptibility, and exposure states
- Adjacency network of individuals

We re-implemented the PALS data generation process in Python and describe the methods here.

Spreader Variables

To draw spreader variables, we rely on 4 parameters - N, the number of individuals, $p_{spreading}$, the proportion of individuals that are spreaders, D_{C+}, the number of variables that are positively correlated with spreader status, and D_{C-}, the number of variables that are negatively correlated with spreader status. We utilize the following steps to draw spreader states:

1. Draw spreader state, c_i, for individual i as $c_i \sim \text{Bern}\left(p_{spreading}\right)$.
2. If an individual has $c_i = 1$, that is, they are a spreader:
   (a) Draw $x_{i,m+} \sim \text{Bern}\left(1/D_{C+}\right)$.
   (b) Draw $x_{i,m-} \sim \text{Bern}\left(0.1/D_{C-}\right)$.
3. Else, if an individual has $c_i = 0$, that is, they are not a spreader:
   (a) Draw $x_{i,m+} \sim \text{Bern}\left(0.1/D_{C+}\right)$.
   (b) Draw $x_{i,m-} \sim \text{Bern}\left(1/D_{C-}\right)$.
4. Concatenate $x_{i,m+}$ and $x_{i,m-}$, to form $x_{i,m}$, the spreader variables for an individual.

In steps 2(a) and 3(b) we set the Bernoulli distribution parameter so that, on expectation, an individual will draw $1/D_C+$ traits. In steps 2(b) and 3(a) we use 0.1 as a constant to essentially minimize the likelihood of an individual drawing those traits. This was arbitrarily chosen and use of a similarly small constant would not significantly impact the outcome.

Susceptibility Variables

The process of drawing susceptibility variables is nearly identical to that of drawing spreader variables, but with different parameters. For susceptibility, we utilize N, the number of individuals, $p_{susceptible}$, the proportion of individuals that are susceptible, D_{S+}, the number of variables that are positively correlated with susceptibility status, and D_{S-}, the number of variables that are negatively correlated with spreader status. We utilize the following steps to draw susceptibility states:

1. Draw susceptibility state, s_i, for individual i, as $s_i \sim \text{Bern}\left(p_{susceptible}\right)$
2. If an individual has $s_i = 1$, that is, they are susceptible:
   (a) Draw $x_{i,l+} \sim \text{Bern}\left(1/D_{S+}\right)$
   (b) Draw $x_{i,l-} \sim \text{Bern}\left(0.1/D_{S-}\right)$
3. Else, if an individual has $s_i = 0$, that is, they are not susceptible:
   (a) Draw $x_{i,l+} \sim \text{Bern}\left(0.1/D_{S+}\right)$
   (b) Draw $x_{i,l-} \sim \text{Bern}\left(1/D_{S-}\right)$
4. Concatenate $x_{i,l+}$ and $x_{i,l-}$, to form $x_{i,l}$, the susceptibility variables for an individual.

Adjacency Matrix

Developing the adjacency matrix is an important part of the data generation process, since we want control over how spreaders are placed through the network. If disease spreaders are scattered through a hospital uniformly, then different exposure and infection outcomes would be expected than if spreaders were highly concentrated in a few wards.

The data generation process utilizes a *stochastic block model* (SBM) - a generative process for random graphs.

We utilize the following parameters - group size, that is, how large each 'group' in the network will be, p_{conc}, a measure of how concentrated spreaders are in the network, and p_{in} and p_{out}, which control the likelihood of edge formation between members of the same and different groups, respectively.

We begin by assigning individuals to groups using the following process.

1. Create $\frac{N}{group_{size}}$ groups, each of which could be thought of as a hospital ward.

2. Assign half the groups to be 'spreader' groups, and the other half of groups as 'non-spreader' groups

3. For each individual that is a known spreader, with a probability of p_{conc}, assign them to a 'spreader group' at random.

   (a) Draw the probability of being assigned to a spreader group, $spr_{group} \sim \text{Bern}\left(p_{conc}\right)$

   (b) If $spr_{group} = 1$, randomly select a 'spreader' group, and assign the individual to that group.

   (c) Else, randomly select a 'non-spreader' group to assign the individual t.

4. For each individual that is not a spreader, we simply randomly assign them to the the remaining spots left in groups.

After groups have been assigned, we have a set of communities, and now draw edges between the communities to create the graph. **For every pair of individuals, i and j:**

1. If i and j are in the same group, draw **edge** $\sim \text{Bern}\left(p_{in}\right)$.

2. Otherwise, if i and j are not in the same group, draw **edge** $\sim \text{Bern}\left(p_{out}\right)$.

3. If **edge** $= 1$, then assign an edge between the two, otherwise, there is no edge between them.

Through these steps we have assigned edge assignments between every pair of individuals, constituting a complete adjacency network.

Exposure States

From the previous steps, we have z, a vector of dimension $(N \times 1)$ with the spreader states of each individual, and adj, the adjacency network of edges between individuals, with dimension $(N \times N)$. We concatenate z by itself N times on the vertical axis, to produce an $(N \times N)$ matrix, z_{full}. Next, we element-wise multiply z_{full} and adj, to produce another $(N \times N)$ matrix, $neighbor_{states}$. Each row in this matrix represents the spreader states of a given individual's neighbors. If two individuals aren't connected, adj has value 0 at $adj(i, j)$, so $neighbor_{states}$ would also have value 0. Consequently, the only values of 1 in $neighbor_{states}$ are when two individuals are connected and one is a spreader.

The PALS model utilized a *mean* influence function, which we will describe here. Further influence functions are described later in this text. The mean influence function simply assigns an *exposed* state to an individual if at least half of their neighbors are spreaders. This produces our exposure vector, η, of size N x 1.

Infection States

Having produced individual susceptibility states and exposure states, we can finally draw the infection states for each individual. We begin by drawing a set of weights, w, which we will use to multiply individual characteristics by to produce the infection states.

Generating w relies on the following parameters - $p(y|S+)$, $p(y|S-)$, and $p(y|\epsilon)$. These parameters control the probability of infection from having a positive susceptibility variable, a negative susceptibility variable, and from exposure, respectively.

1. We draw the non-exposure-based infection weights as the following:

    (a) $w_{S+} = N(\sigma^{-1}(p(y|S+)), 0.1I)$

    (b) $w_{S-} = N(\sigma^{-1}(p(y|S-)), 0.1I)$

2. We draw the exposure-weights in a similar fashion. $w_e = N(\sigma^{-1}(p(y|\epsilon)), 0.1I)$

Note that here, N is a multivariate Gaussian distribution. w_{S+} is a matrix of size $(N \times D_{S+})$, w_{S-} is a matrix of size $(N \times D_{S-})$, and w_{ϵ} is a vector of size $(N \times 1)$. I simply represents the identity matrix, utilized for covariance. σ^{-1} represents the inverse sigmoid function $y = log(x/(1-x))$.

At this point we may concatenate the three matrices above, to form w, our matrix of infection weights. We similarly concatenate $x_{i,l}$ and η, to produce x, the matrix of individual characteristics. Finally, we draw the infection states:

1. $Inf_{mult} = w\ x$

2. $Inf_{sigmoid} = \sigma(Inf_{mult})$

3. $y = \sim \text{Binomial}\,(Inf_{sigmoid})$

Inf_{mult}, $Inf_{sigmoid}$, and y will all be of shape $(N \times 1)$. σ represents the sigmoid function, defined as $y = 1/(1+e^{-x})$.

Modifications to PALS Data Generation

While the PALS data generation method described above largely produces the type of synthetic data we desire, there were minor tweaks made to improve data generation. Consider a simple example, with D_{S+}, $D_{S-} = 1$. We use this example to further elucidate the PALS data generation method. The infection weight vector will be of shape (3×1), and assume probabilities $p(y|E) = 0.8, p(y|S+) = 0.3, p(y|S-) = 0.1$. Through the weight generation process, the weight vector, on expectation, will be will be $< -0.847, -2.197, 1.386 >$. Assume that individual characteristic vector is of format <*Positive Susceptibility Trait*, *Negative Susceptibility Trait*, *Exposure Trait*>. We begin by defining the following example individuals:

1. $< 1, 0, 0 >$ - an individual exhibiting the positive susceptibility characteristic

2. $< 0, 1, 0 >$ - an individual exhibiting the negative susceptibility characteristic

3. $< 0, 0, 1 >$ - an individual exhibiting the exposure characteristic

4. $< 0, 0, 0 >$ - an individual exhibiting no characteristics

5. $< 0, 1, 1 >$ - an individual exhibiting the exposure characteristic and the negative susceptibility characteristic

6. $< 1, 0, 1 >$ - an individual exhibiting the exposure characteristic and the positive susceptibility characteristic

Next, we calculate the infection state of each individual, as described in the previous section, by multiplying individual characteristics by the weights and applying a sigmoid function to the output.

1. $\sigma(< 1, 0, 0 > * < -0.847, -2.197, 1.386 >) = 0.3$

2. $\sigma(< 0, 1, 0 > * < -0.847, -2.197, 1.386 >) = 0.1$

3. $\sigma(< 0, 0, 1 > * < -0.847, -2.197, 1.386 >) = 0.8$

4. $\sigma(< 0, 0, 0 > * < -0.847, -2.197, 1.386 >) = 0.5$

5. $\sigma(< 0, 1, 1 > * < -0.847, -2.197, 1.386 >) = 0.3$

6. $\sigma(< 1, 0, 1 > * < -0.847, -2.197, 1.386 >) = 0.63$

The first three results correspond to the probability parameters defined above. However, the issues present with the sigmoid-based probabilities manifest themselves in the individuals 4 and 6. To begin, individual 4 has a 50% chance of infection, even though they are neither susceptible nor exposed. Additionally, individual 6 is both susceptible *and* exposed, yet has a lower probability of infection than individual 3, who is only exposed.

As a result, we make the following modifications to the data generation process.

- If an individual has a characteristic vector of all 0's - $< 0, 0, 0, ..., 0 >$ - they are not exposed, and their susceptibility state is unknown based on this vector. Since we assign a ground truth susceptibility state for every individual in the data generation process, we assign infection probability based on this. If the individual was designated as susceptible, we assign the infection probability as $p(y|S+)$ and otherwise $p(y|S-)$. This ensures these individuals do not have the artificially large infection probability of 0.5.

- If an individual has a states vector with positive susceptibility traits and the exposure trait, then we set their probability of infection to be the minimum of their current state, $p(y|S+)$ and $p(y|E)$. This ensures the probability is never lower than it should be.

3.1.2 Hospital Data

To perform validation on a real-world dataset, we use data from a large urban hospital, with specific criteria as mentioned in Chapter 2. We utilize a super-set of the cohort in the PALS work, since we were unable to apply the exclusion criteria mentioned in thee original research paper [8]. The cohort of patients ended up comprising 23,889 individuals, with 1292 individual variables. The roommate cohort included 47,356 patients, with 1772 individual variables.

3.2 NeuralPALS

The model we develop in this work is a neural network to learn latent spreader, exposure, and infection states. We begin by defining the inputs to the model - $x_{neighbors}$, x_{cohort}, adj, and y_{cohort}.

- $x_{neighbors}$ - $(n_{neighbors} \times D_{neighbors})$ matrix, with $D_{neighbors}$ being the individual features of the neighbors. As mentioned, in the MGH data, the spreaders are not in the cohort themselves, so we use $x_{neighbors}$ to learn spreader states

- x_{cohort} - $(n_{cohort} \times D_{cohort})$ matrix, similar to $x_{neighbors}$, but for cohort members.

- y_{cohort} - $(n_{cohort} \times 1)$ matrix the infection labels for members of the cohort

- adj - $(n_{cohort} \times n_{neighbors})$ matrix, an adjacency network between the cohort and neighbors.

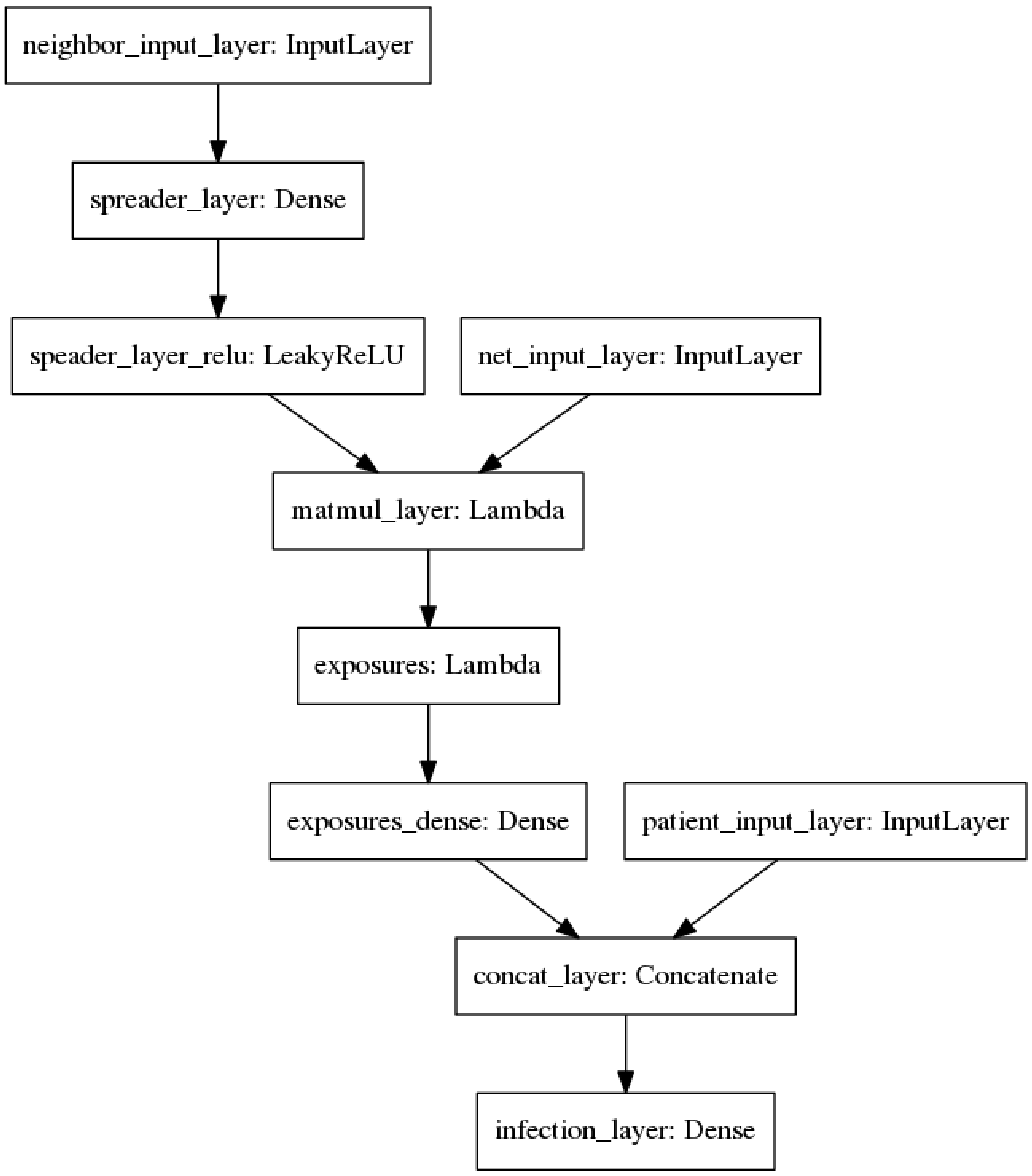
neighbor_input_layer: InputLayer
spreader_layer: Dense
speader_layer_relu: LeakyReLU
net_input_layer: InputLayer
matmul_layer: Lambda
exposures: Lambda
exposures_dense: Dense
patient_input_layer: InputLayer
concat_layer: Concatenate
infection_layer: Dense

Figure 3-1: NeuralPALS Architecture

The first section of the model learns the spreader states. The input to this section is $x_{neighbors}$, and we connect it to a fully-connected (*Dense*) layer with 1 output neuron, which is subsequently connected to an activation layer. We choose a LeakyReLU activation because of its efficacy at resolving the vanishing gradient and dying ReLU problems [5]. The outputs of this layer are theoretically the spreader states for the model. The fully-connected layer and activation functions should allow the model flexibility in learning non-linear functions.

The next section of the model is calculating the exposure states. As mentioned previously, there are multiple ways to conceptualize learning the exposure states. A natural approach to this problem would be the following - we consider each cohort member's neighbor's spreader state by building a $(n_{cohort} \times n_{neighbors})$ matrix, and connect this to a fully-connected layer, achieving an output size of $(n_{neighbors} \times 1)$. This output layer would theoretically be the exposures. However, this approach is flawed. Traditional neural network methods often treat input as vectors, with different columns representing different variables. However, in our example, the location of a spreader in the matrix doesn't matter - that is to say, our learning problem is *permutation invariant.* We define permutation invariance in Chapter 2. As a result, we build a permutation invariant layer, as inspired by the work of Zaheer et al. [7].

We have mapped spreader states to a latent dimension through the use of the fully connected layer and activation function at the end of the first section of the model. We take advantage of the finding that summation is a permutation invariant in the latent space [7]. First, we multiply the adjacency matrix by the spreader states, producing a matrix of shape $(n_{cohort} \times 1)$. This serves as summation in the latent space. We then use a fully-connected layer to learn potential non-linear functions on the output of the matrix multiplication layer. The output of this layer can be understood as the exposures.

The final section of the model parallels the first. We concatenate the newly learned exposures to x_{cohort}, producing a matrix of size $(n_{cohort} \times (n_{neighbors} + 1))$. We then connect a fully connected output layer with 2 neurons - the final layer, infection status, can be thought of as a binary classification problem. We utilize a softmax activation layer to normalize output into a probability distribution, representing probability of infection status.

3.3 Influence Functions

As mentioned previously, the influence function present in the spread of a contagion is unknown. As a result, one of the goals of NeuralPALS was to develop a model that is agnostic to different influence functions, or at the least is easily adaptable to them. Assume z is the vector of learned spreader states, and adj is the adjacency network of individuals. When considering influence functions we ignore how z is learned, instead focusing on determining the impact of z. We also define η as the exposure states vector, the output of each influence function.

A simple influence function utilized by previous work is simply the **mean** influence function, defined as $\eta_i = 1$ if $\frac{\sum_{j=1}^{N_{neighbors}} z_j}{N_{neighbors}} \geq 0.5$ else 0, where $N_{neighbors}$ are the neighbors of i, that is, individuals with which i shares an edge. While this influence function excels in its simplicity, its not entirely representative of real-world infection spread. For example, an individual walking around a mall with 500 people, of which 100 are spreaders, is classified as not-exposed, whereas an individual who has three neighbors, of which two are spreaders, is classified as exposed. For many diseases, being in the presence of non-spreaders does not confer additional protection against a disease.

As a result, a logical alternative influence function would be a **thresholded influence function**, defined as $\eta_i = 1$ if $\sum_{j=1}^{N_{neighbors}} z_j \geq \alpha$ else 0, where α is the threshold. This provides a magnitude based influence function.

For highly viral infections, sometimes exposure to just 1 spreader is enough for an individual to be considered exposed. As a result, we define a specific case of the threshold influence function, as the **minimum influence function**, formally defined as $\eta_i = 1$ if $\sum_{j=1}^{N_{neighbors}} z_j \geq 1$ else 0.

While unlikely in infection spreading settings, it is possible that every neighbor of an individual must be spreading, leading to the **maximum influence function**. This is similar to the aforementioned mean function, but $\eta_i = 1$ if $\frac{\sum_{j=1}^{N_{neighbors}} z_j}{N_{neighbors}} = 1$ else 0.

We define all of these influence functions in our data generation method and synthesize data with differing influence functions to test models' robustness to different influence functions.

3.4 Baselines

In order to compare our methods we utilize numerous baselines. We include baselines from previous work. We choose these baselines to isolate various aspects of learning.

3.4.1 NoNet

The simplest baseline we present is one that simply attempts to understand infection state from susceptibility variables. In cases where exposure is irrelevant, it is expected that this baseline, NoNet, would perform well. Conversely, when exposure is a significant aspect of infection status, NoNet would be expected to perform poorly. We implement NoNet as a logistic regression, learning weights from susceptibility variables to infection status.

3.4.2 z_0

This baseline builds off NoNet by including both spreader variables and susceptibility variables. z_0 is provided with spreader states at training time, and so attempts to learn a weight mapping from spreader variables to spreader states for use at test time. We implement z_0 as two logistic regressions, to learn spreader and infection variable weights.

3.4.3 n_0

This baseline presents the best-case scenario for learning, in which spreader states, and as a result, exposure, are fully known during both training and testing time. As a result, we do not have to develop a logistic regression to learn spreader states, and instead only train a regression on the infection states.

3.4.4 n_{0k}

While some baselines include all of the spreader states for training, testing, or both, it's unlikely in the real world that every spreader state is known. As a result, comparing partial spreader training to a baseline is important. n_0k is trained on a subset of the spreader labels in training time, and then attempts to predict spreader and infection states at test time.

3.4.5 PALS

Finally, we compare our model to the original PALS model, described in detail in Chapter 2. We utilize inference code provided by the authors to provide a faithful recreation.

3.5 Experimental Setups

We employ a series of experiments to verify the effectiveness of NeuralPALS. We investigate learning spreader states, exposure states, and infection states through various data generation parameters. Full experiment details and results are presented in Chapter 5. We primarily utilize AUC as our evaluation metric. Since we frequently work with imbalanced datasets, simple binary accuracy does not suffice as an evaluation metric - a model could overfit and learn one class and exhibit very high accuracy in an imbalanced dataset.

Chapter 4

Experiments and Results

In this chapter we present results on our experiments with synthetic data and hospital data. We begin by defining our default data generation parameters and then explain experiments run with NeuralPALS on the synthetic data. Then, we explain how we extended our model to a real-world dataset and present preliminary findings.

4.1 Experimental Setup

We utilize the following parameters in our experimental setups, unless otherwise specified. All variable descriptions are present in Chapter 3.

- $n_{train} = 500$
- $n_{test} = 500$
- $p_{in} = 0.5$
- $p_{out} = 0.01$
- group size $= 10$
- $p_{conc} = 0.9$
- $p_{spreading} = 0.5$
- $D_{C+} = 5$, $D_{C-} = 5$

- $p_{susceptible} = 0.5$
- $D_{S+} = 5$, $D_{S-} = 5$
- $p(z|C+) = 0.8$
- $p(z|C-) = 0.2$
- $p(y|S+) = 0.5$
- $p(y|S-) = 0.5$
- $p(y|E) = 0.9$
- Exposure Function = **mean**

4.2 Understanding The Impact of Exposure

In this experiment we vary the probability of infection given exposure, while keeping the probability of infection given susceptibility constant at 0.5. This experiment essentially isolates the impact of exposure on infection.

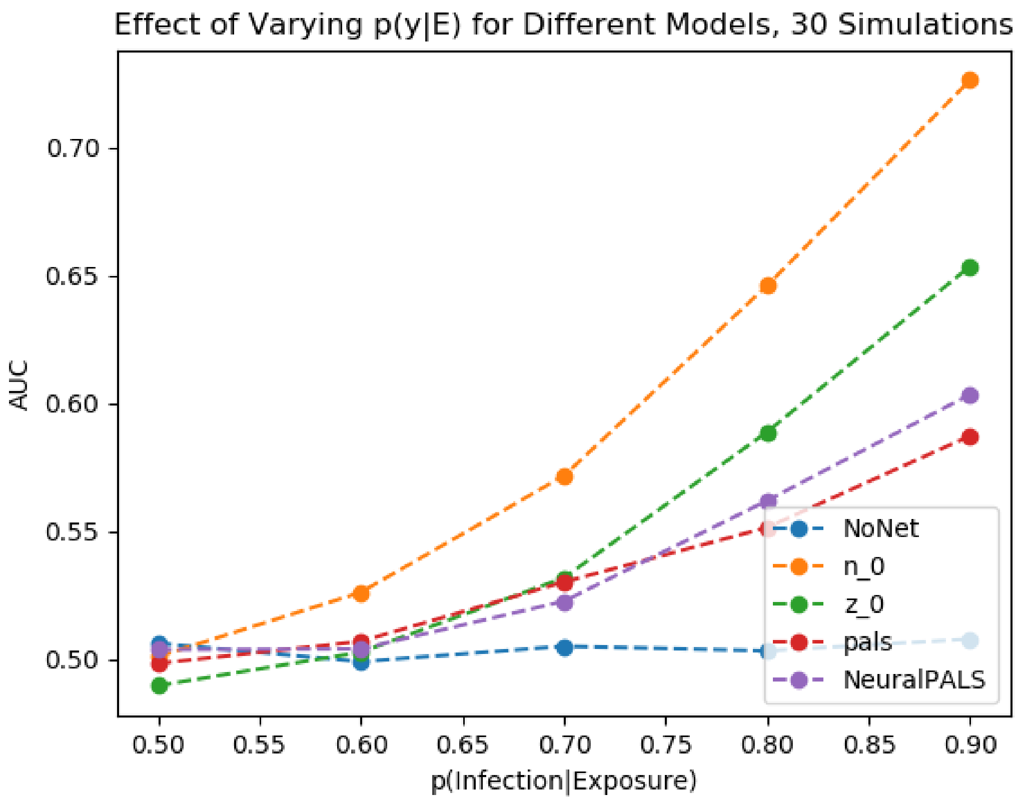

Figure 4-1: Effect of Varying $p(y|E)$ on Infection AUC

The result of this experiment are shown in Figure 4-1. As expected, NoNet maintains an AUC around 0.5 regardless of $p(y|E)$. This is expected - NoNet does not have information or predictive power of exposure, so an AUC of 0.5, exhibiting no predictive power, is reasonable. In general, all of the other models increase their predictive power as $p(y|E)$ increases. The oracles perform the best - n_0 is provided with all spreader states (train and test), so it is expected to exhibit the best performance. z_0 receives access to the spreader states at training time, leading to its second-best performance. NeuralPALS performs comparably to PALS throughout the experiment.

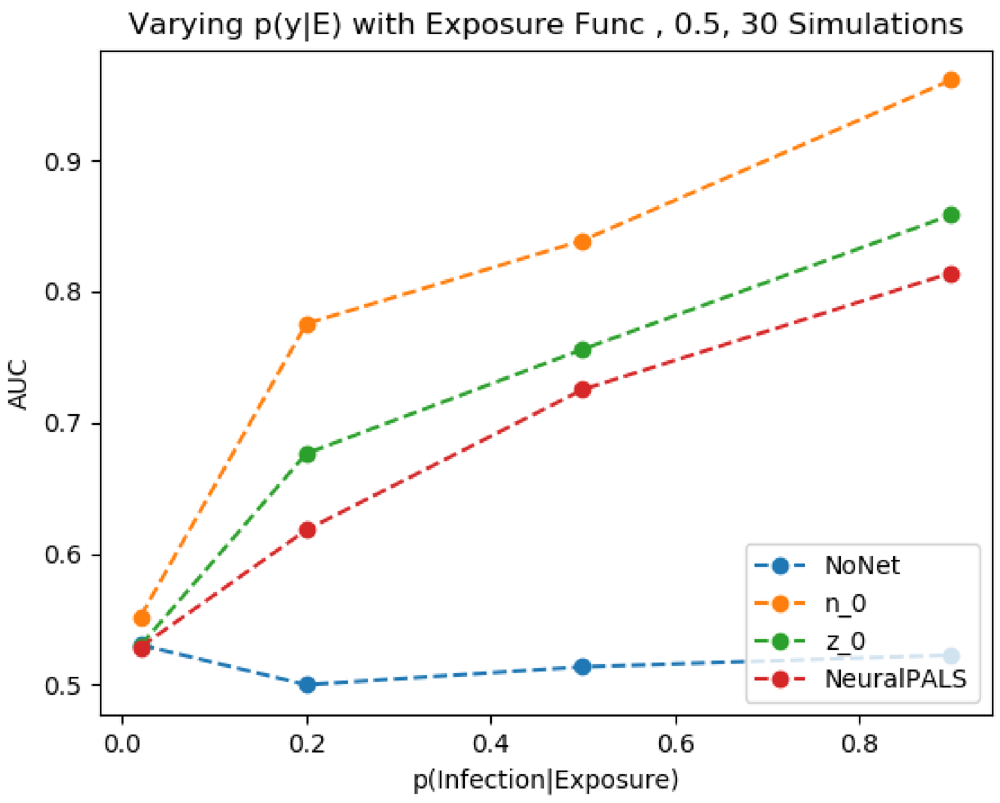

Figure 4-2: Effect of Varying $p(y|E)$ on Infection AUC

Next, we set $p(y|S+) = 0.02$ and $p(y|S-) = 0.002$. As a result, few individuals should be infected exclusively from susceptibility without exposure. This experiment further isolates the effect of exposure on infection state. Unfortunately, we were unable to run the PALS baselines for multiple experiments because of issues with the original codebase. The observed trends of this experiment are shown in Figure 4-2. We vary $p(y|E)$ in this experiment from 0.02 to 0.9. The result of this experiment are similar to those in 4-1, and NeuralPALS is quite close in AUC to z_0, even though NeuralPALS is not trained on spreader states. This indicates that NeuralPALS is able to learn spreader states without direct access to them at training or test time.

4.3 Understanding the Impact of Susceptibility

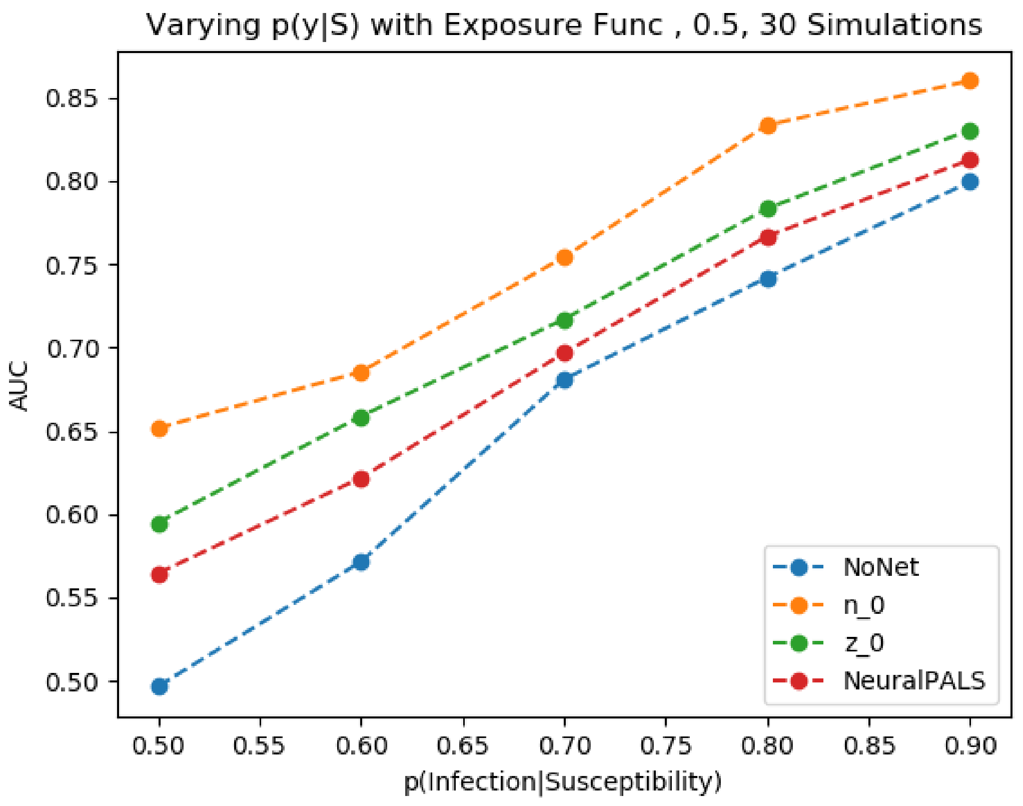

Figure 4-3: Effect of Varying $p(y|S)$ on Infection AUC

In this experiment, we isolate the impact of susceptibility, varying $p(y|S+)$ from 0.5 to 0.9. As expected, all of the models improve their AUC as $p(y|S+)$ is increased. When $p(y|S+)$ is less than 0.7, it is clear that the slight impact of exposure separates NeuralPALS and the oracles from NoNet, but once susceptibility is the primary cause of infection then all of the models converge.

4.4 Learning Exposure States

Effectively learning exposure states is also an important component of NeuralPALS. We set up multiple trials and examine NeuralPALS' effectiveness at learning the exposure states.

Exposure Function	$p(spreading)$	$p(y\|E)$	$p(y\|S+)$	$p(y\|S-)$	AUC
Mean	0.5	0.9	0.9	0.2	0.649
Mean	0.5	0.9	0.2	0.2	0.759
Max	0.7	0.9	0.9	0.2	0.602
Max	0.7	0.9	0.2	0.2	0.697

We include two exposure functions - the **mean** and **max** influence functions. We vary $p(y|S+)$ to modify the difficulty of the task. The $p(y|S+) = 0.2$ setting is designed to be the easier task - the majority of infected patients must be infected through exposure instead of susceptibility. NeuralPALS achieved greatest exposure AUC on the mean exposure function, with $p(y|S+) = 0.2$, the easier difficulty of the two $p(y|S+)$ settings. Additionally, the mean influence function is a function that could be effectively modeled by a traditional regression, so its unsurprising that NeuralPALS could model it effectively. NeuralPALS was able to attain AUCs on the max influence function that were on average only 5% worse than on the mean influence function. The max function is a non-linear influence function and this finding demonstrates NeuralPALS' potential to effectively learn this class of functions.

4.5 Hospital Data

As mentioned previously, we use a dataset of over 70,000 patients. We split the data into test and train networks by year - patients hospitalized from May 2012 - May 2013 are in the training set and patients hospitalized from May 2013 - May 2014 are in the test set. This mimics the approach taken by the original PALS work. The dataset is highly imbalanced - 1.61% of the training cohort is infected and 1.58% of the test cohort is infected. We utilize the roommate network, constructed through the methods described in Chapter 3.

4.6 NeuralPALS on Real Data

We train the NeuralPALS model on the data described above. To account for the imbalanced dataset, we implement *sample weighting* - in our loss function, samples of the infected class are weighted approximately 60x more than uninfected samples, to prevent the model from overfitting to the uninfected label class. We train the model for 100 epochs using the *Adam* optimizer.

4.7 Real Data Experiment

At this stage of the work, we have recently begun experiments on the real data, so the preliminary results are very limited. We implement a logistic regression as a simple baseline to understand if NeuralPALS is learning the impact of exposure on infection, instead of just the impact of susceptibility. We do not implement n_0 or z_0, the baselines from our synthetic data experiments, as we do not have ground truth access to the spreader state labels.

Model	Infection AUC (averaged over 20 trials)
Logistic Regression Baseline	0.55
NeuralPALS	0.643

These experiments show increased predictive power using NeuralPALS as compared to the baseline - theoretically the impact of exposure. We detail the many directions of future work present in Chapter 5.

Chapter 5

Conclusions and Future Work

5.1 Summary

In this work we developed NeuralPALS, a neural network-based approach to understanding contagion spread. Specifically, we focused on identifying spreader, exposure, and infection states. NeuralPALS strove to utilize permutation invariant architecture to learn exposure functions for contagion spread. We incorporated synthetic data generation to validate the various goals of our research. In all synthetic experiments, NeuralPALS was shown to perform superior to a baseline, NoNet, that did not incorporate exposure to learn infection states. NeuralPALS was also shown to perform comparably to PALS, the work which much of this work was built off of. Experiments were performed to demonstrate NeuralPALS's ability to learn exposure states with different exposure functions, demonstrating the flexibility of the model. We also presented preliminary results on a dataset from a large urban hospital, with the task of predicting C. diff infections. NeuralPALS once again demonstrated improved predictive power over a baseline that doesn't account for exposure state. We hope that the NeuralPALS model may be utilized to learn unknown exposure functions for various infectious diseases.

5.2 Limitations

Certain limitations of our approach must be acknowledged. Comprehensive comparison to the original PALS model could not be performed for multiple experiments due to problems with implementing the PALS inference. When working with the real-world data we ran into multiple limitations as well. We excluded members of the cohort from the spreading population - a form of selection bias that we hope to correct for in future work. In addition, we were unable to duplicate the exact cohort exclusion criteria used in the PALS research, preventing a direct comparison of model accuracies. For both the synthetic and real-world experiments we assume a simplistic network structure - that edges are one-way and unweighted. This is a limited representation of true contagion spreading mechanisms.

5.3 Future Work

There are numerous avenues for future work. To begin, we must perform a complete comparison to PALS to fully understand drawbacks and advantages of the NeuralPALS model. The PALS work also presented results on partial-spreader training. This is when the spreader states of a few members of the population are known at training and potentially testing time. Understanding NeuralPALS's ability to extrapolate from partial spreader knowledge is vital in our comparisons. Furthermore, in this work we only present preliminary results on the real-world hospital dataset. Detailed experimentation to improve infection prediction accuracy on this dataset still needs to be performed. In this work we only analyzed the roommate network and experiments still need to be conducted on the nursemate network.

An important avenue for future work is developing more advanced network structures. Incorporating weighted edges to account for time spent together by neighbors could allow for a better understanding of exposure. In addition, combining the nursemate and roommate networks could help understand the relative impacts of each on exposure.

Our model assumes a stationary view of contagion spread. Adding a temporal aspect to NeuralPALS would help researchers understand how contagion spread evolves over time.

Bibliography

[1] Nida Shahid, Tim Rappon, Whitney Berta. Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PLoS ONE*, February 2019.

[2] Fernanda C. Lessa, Yi Mu, Wendy M. Bamberg, Zintars G. Beldavs, Ghinwa Dumyati, John R. Dunn, Monica M. Farley, Stacy M. Holzbauer, James I. Meek, Erin C. Phipps, Lucy E. Wilson, ZintarsLucy E. Wilson, et al. Burden of clostridium difficile infection in the united states. *New England Journal of Medicine*, February 2015.

[3] Shelley S. Magill, Erin O'Leary, Sarah J. Janelle, Deborah L. Thompson, Ghinwa Dumyati, Joelle Nadle, Lucy E. Wilson, Marion A. Kainer, Ruth Lynfield, Samantha Greissman, Susan M. Ray, Zintars Beldavs, et al. Changes in prevalence of health care-associated infections in U.S. hospitals. *New England Journal of Medicine*, November 2018.

[4] Zhengming Xing, Bradley Nicholson, Monica Jimenez, Timothy Veldman, Lori Hudson, Joseph Lucas, David Dunson, Aimee K. Zaas, Christopher W. Woods, Geoffrey S. Ginsburg, and Lawrence Carin. Bayesian modeling of temporal properties of infectious disease in a college student population. *Journal of Applied Statistics*, 2014.

[5] Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li. Empirical evaluation of rectified activations in convolutional network. *ICML Deep Learning Workshop*, 2015.

[6] Christopher A. Bail, Lisa P. Argyle, Taylor W. Brown, John P. Bumpus, Haohan Chen, M. B. Fallin Hunzaker, Jaemin Lee, Marcus Mann, Friedolin Merhout, and Alexander Volfovsky. Exposure to opposing views on social media can increase political polarization. *Proceedings of the National Academy of Science of the United States of America*, September 2018.

[7] Manzil Zaheer, Satwik Kottur, Siamak Ravanbhakhsh, Barnabas Poczos, Ruslan Salakhutdinov, Alexander J Smola1. Deep sets. *31st Conference on Neural Information Processing Systems*, 2017.

[8] Maggie Makar, John Guttag, Jenna Wiens. Learning the probability of activation in the presence of latent spreaders. *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.