PhyloMissForest: a framework to construct phylogenetic trees with missing data

Diogo Manuel Barata Pinheiro
Instituto Superior Técnico
Lisbon, Portugal
diogo.pinheiro@tecnico.ulisboa.pt

Abstract—In the pursuit of a better understanding of biodiversity, evolutionary biologists rely on the study of phylogenetic relationships to illustrate the course of evolution. The relationships among natural organisms, often depicted in the shape of phylogenetic trees, not only help to understand evolutionary history but also have a wide range of additional applications in science. One of the most challenging problems that arise when building phylogenetic trees is the presence of missing biological data. More specifically, the possibility of inferring wrong phylogenetic trees increases proportionally to the amount of missing values in the input data. Although there are methods proposed to deal with this issue, their results remain unsatisfactory and lacunose.

This Thesis proposes a framework, called PhyloMissForest, to impute missing entries in phylogenetic distance matrices. PhyloMissForest is built upon a random forest structure that infers the missing cells of the input data, based on the known parts of it. PhyloMissForest contributes with a robust and configurable framework that incorporates multiple search strategies and machine learning techniques, guided by phylogenetic knowledge, to provide a more accurate inference of lost phylogenetic distances.

We evaluate our framework with three real-world datasets, two DNA-based sequence alignments and one containing amino acid data. Moreover, we perform a design of experiments study to define the hyperparameter values of our algorithm, which is a concise method, preferable in comparison to the well known exhaustive parameters search. In the DNA datasets, by varying the percentages of missing data from 5% to 60%, we outperform the state-of-the-art alternative imputation techniques in 100% of the tests. As for the amino acid instance, our framework remains competitive with 50% of wins. PhyloMissForest proved that the merge between machine learning techniques and phylogenetic knowledge provides a valuable tool for phylogenetic research in the presence of missing data.

Index Terms—Phylogenetic Tree, Missing Data, Imputation, Machine Learning, Random Forest

I. INTRODUCTION

The understanding of evolutionary history and relationships among individuals or groups has become a key research topic over the years. This area of study is called phylogenetics, where the relationships are inferred throughout the genetic material. Phylogenetic studies not only help to understand the relationships between organisms but also have a wide range of other important applications. They help the conservation researchers to study and make decisions about the protection of endangered species and conservation policies [1]. The application of phylogenetics also revealed to be important in the forensics sciences [2]. It also improves the investigation of pathogens, also called molecular epidemiology, thus being a useful tool to understand and fight against infectious diseases [3]. The inclusion of phylogenetics in medicine proved to be a tool that significantly improves the detection of complex diseases like cancer [4]. In the area of pharmacology, using phylogenetic trees also helps researchers in drug development [5].

To represent evolutionary relationships, a graphical structure called phylogenetic tree is built. The methodology used to create phylogenetic trees is divided into two major groups: character-based methods and distance-based methods [6]. While the former uses the alignment to generate a set of possible phylogenetic trees directly from the sequences at hand, the latter uses the given sequences to construct a phylogenetic pairwise distance matrix instead and then from the matrix construct a phylogenetic tree. The most popular character-based methods are: maximum parsimony, maximum likelihood and Bayesian methods. In terms of the distance-based methods, the ones that are mostly used are: UPGMA, WPGMA, NJ and FM. When using large-scale datasets, character-based methods are computationally demanding. In contrast, distance based methods are typically less computationally demanding, which is crucial for fast and accurate phylogenetic analysis [7]. Moreover, there are evidences that distance-based methods have competitive and faster results when comparing with character-based methods [8]. Therefore, distance-based methods will be a point of focus during this Thesis.

Distance-based methods are composed of two major steps: 1) computing the distance matrix and 2) obtaining the phylogenetic tree from the distance matrix. In order to perform the second step, the pairwise distance matrix, obtained in the first step, needs to be complete. In order to perform the second step, the pairwise distance matrix obtained in the first step needs to be complete, i.e, all its values should be known. However, in modern phylogenetic studies, it is common that not all values in distances matrices are known (i.e, it has missing data), which makes the process of building a phylogenetic tree a very difficult task. In fact, with the increase in the amount of data in a study, the probability of missing data also increases [9]. Missing data in the phylogenetics field may occur for several reasons: failure of experimental work [7], data generation protocols, approaches to taxon and gene sampling and gene birth and loss [10]. Missing data can also appear due to the lack of biological material, imprecision of experimental methods and for a combination of unpredictable...
Aiming to build the pairwise distance matrix, which is required to construct a phylogenetic tree using one of the distance-based methods, the first step to perform is the acquisition of the sequences. Having the sequences of data, the second step is the alignment of the sequences. Finished the process of sequence alignment, now the dissimilarity between each sequence is calculated, i.e., when comparing two sequences, each column is analysed and the distance between them is the number of different columns they share. There are several criteria of dissimilarity, for example, Hamming Distance. The pairwise distance matrix described is constructed from the alignment of the sequences. Hence, the problem of missing data in phylogenetic distance matrices arises from the missing characters in the sequence alignment [12]. In Figure 1, there is an example of a distance matrix obtained from an alignment with missing characters. Note that the example illustrated in Figure 1 is about DNA genetic sequences, that are composed of 4 nucleobases: A-adenine, C-cytosine, G-guanine and T-thymine.

![Fig. 1. Pairwise distance matrix obtained from incomplete sequences.](image)

In order to compute the distance between two sequences, after aligned, they have to share a known part between each other. Given that, by analysing Figure 1, we can conclude that the distance between sequence B and C cannot be computed, since they do not share any known part. In other words, in the part sequence B has known characters, sequence C has missing characters, represented with the question mark, and the vice versa.

Studies revealed that with the presence of missing data in phylogenetic analysis, the possibility of inferring wrong phylogenetic trees also increases [13], [14]. Therefore, there is a need of efficient strategies to deal with missing data in phylogenetic pairwise distance matrices. The development of those strategies is a research topic with a short development. Even so, we can divide the existent approaches in two main groups: direct and indirect methods. The former infer the phylogenetic tree directly from the partial distance matrix, while the latter, first implements a strategy to fill the missing information in the distance matrix and after that, construct the phylogenetic tree from the imputed matrix.

II. RELATED WORK

Starting from the direct strategies, which are the ones that infer the phylogenetic tree directly from the incomplete distance matrix, the existent methods are: an approach based on the triangles method [15], a LS-based method named MW-modified [11] and an adaptation of NJ to deal with incomplete distance matrices [16]. To be possible to apply the referred methodologies, a combination of restrictions need to be addressed. In the triangles method, to add a new element to a tree, it needs to share at least two known distances with elements within the tree. MW-modified assumes that the phylogenetic tree needs to be active, meaning that the distances in the matrix need to correspond to the distances in the phylogenetic tree, which in practice is not always true. Furthermore, when the percentages of missing data increase, these methods are not scalable enough. The combination of this fact with the restrictions imposed by each method, limits the application of the existent approaches, making them limited and unsatisfactory.

In terms of the indirect strategies, which are the methods that first infer the missing distance and then from the imputed distance matrix construct the phylogenetic tree, the existent methods are: an heuristic approach named Lasso [7], a LS-based approach with multivariate optimization called DAMBE [12], a statistical method called SIA [17] and two machine learning techniques: matrix factorization and autoencoder [18].

To apply Lasso, the molecular clock hypothesis needs to be assumed, meaning that the divergence accumulates at a constant rate, which in practice is a constraint difficult to be ensured. DAMBE proved that it is possible to build a phylogenetic tree without assuming the molecular clock hypothesis. However, this method cannot ensure the phylogenetic reconstruction with more than 10% of missing data, which is also the main issue with SIA method. In terms of the matrix factorization and autoencoder approaches, when testing their implementations we notice that both methods allow the final matrices to have negative values. In a phylogenetic perspective, negative distances are meaningless and therefore, these distances do not respect the properties of a phylogenetic pairwise distance matrix [19]. Moreover, the execution time of the matrix factorization approach is way to slow.

Given all the problems identified in the current state-of-the-art methodologies and since their results remain unsatisfactory, the need of efficient strategies to deal with missing data in phylogenetic distance matrices remains an open problem with no accurate method to deal with.

III. PHYLOMISSFOREST

Across this Thesis, to be possible to construct a phylogenetic tree from a partial distance matrix, we developed a ML-based framework. To accomplish this task, we took as a reference an imputation method called MissForest present in [20], which is a random forest based imputation algorithm for
every type of missing data. Aiming to guide it for our problem, we developed a number of search techniques, coupled with phylogenetic knowledge and criteria, to accurately combat the missing data problem.

A. MissForest Review

As it was previously mentioned, our framework is built upon MissForest algorithm. Therefore, after exposing the strategies proposed in our framework, it is crucial that the idea behind the baseline MissForest is well understood. Hence, during this subsection, a brief explanation on the behave of the referred methodology is introduced. The main idea behind MissForest is to train successive random forests with the known parts of the dataset and use the trained random forest to predict the unknown values. The pseudo-code of the algorithm adapted from [20] is present in the Algorithm 1.

**Algorithm 1 MissForest algorithm**

0: procedure MissForest(M);
1: Create a boolean mask to mark the position of the missing values
2: Initial guess of the missing values - mean value of each column
3: Vector k of missing values per column
4: old\_error = \infty; new\_error = 0; M\_old = M\_new = M
5: while old\_error \geq new\_error do
6: Update old\_error and new\_error if not iteration 0
7: Update M\_old
8: for c in k do
9: Fit a random forest with X\_obs and y\_obs
10: Predict y\_miss with X\_miss
11: Update M\_new with y\_miss
12: end for
13: Calculate the new\_error
14: end while
15: Return M\_old

In order to better understand the behaviour of the algorithm, some variables have to be defined:

- \( M \) is the matrix inputted by the user;
- \( M\_old \) and \( M\_new \) are the matrices from the previous and actual iterations, respectively;
- \( old\_error \) and \( new\_error \) are the errors from the previous and actual iterations, respectively;
- \( X\_obs \) and \( y\_obs \) are the variables used to train the forest;
- \( X\_miss \) is the variable used by the trained forest in order to predict \( y\_miss \)

In the first step of the process, in order to know where the missing values are, the algorithm creates a boolean mask with the same dimensions of the inputted matrix \( M \). The positions set to the logic value of true are the unknown, while the positions where the logic value is false refer to the known entries in the input. Finished the creation of the mask, an initial guess for the missing values is calculated. Taking into account a specific column, the missing values are imputed with an initial guess that corresponds to the mean of this column. This process is repeated for every column in the dataset. The algorithm counts the number of missing values per column, in order to set the order in which the imputation will occur. By default, the imputation starts in the column with less missing values. In case of a tie in the number of missing values, the column with lower index among the columns that are tied, is taken first. The initial values of \( old\_error \) and \( new\_error \) are \( \infty \) and 0, respectively. This is to force the algorithm to run at least two iterations. Therefore, the first iteration is considered as a zero iteration. \( M\_old \) and \( M\_new \) are set with a copy of the input matrix \( M \).

Given the initial process as defined, the imputation loop begins. The initial step of the referred loop is the update of the matrix \( M\_old \) with a copy of the matrix \( M\_new \). This is to update the matrix of the previous iteration before the new iteration begins. In iteration zero, this step is ignored because the initialization explained above substitutes this step. After the update of the \( M\_old \), the algorithm starts the loop that iterates over all the columns by the order defined in the vector described above. During the execution of this loop, for each column, the algorithm defines the following variables: \( X\_obs \), \( X\_miss \), \( y\_obs \) and \( y\_miss \). A random forest model is fitted with \( X\_obs \) and \( y\_obs \), given as an input. The model already trained uses the predicting function in order to obtain \( y\_miss \) from the given \( X\_miss \). At the end of each iteration of this loop, the predicted values \( y\_miss \) are imputed in the unknown positions of the column \( c \), which is taken into account in this iteration. This process is repeated for every column in the dataset.

Finished the imputation of all the columns, the error between \( M\_old \) and \( M\_new \) is calculated by applying equation (1). The previous \( new\_error \) is copied to the \( old\_error \) and the \( new\_error \) obtains a new value.

\[
new\_error = \frac{\sum_{i,j}(X\_new - X\_old)^2}{\sum_{i,j}(X\_new)^2} \tag{1}
\]

In case the \( new\_error \) is smaller than the \( old\_error \), the algorithm moves to another iteration. Otherwise, the loop stops and the algorithm returns the imputed matrix from the previous iteration, \( M\_old \), since it gives a smaller error than the \( M\_new \).

B. PhyloMissForest Structure

As it was explained, MissForest is an imputation algorithm based on random forests that is able to infer any type of missing data. Since our type of missing data is always continuous and our objective is to reconstruct phylogenetic trees from partial pairwise distance matrices, we should change the decisions of the algorithm in order to guide it for our goal.

In Figure 2, there is a flow chart of each phase of the PhyloMissForest. Above each phase, there is a description of the process introduced in PhyloMissForest that forces our methodology to behave in phyloinspired manner:

**Initial guess** - In the initialization step of the algorithm, PhyloMissForest introduces a new technique to ensure that after the initial guess of the missing values, the matrix respects the properties of a phylogenetic distance matrix;

**Splitting tie-break criteria** - During the imputation cycle of the PhyloMissForest, our framework introduces a methodology to deal with the ties during the splitting process in the decision trees building. Here our framework not only supports the default configuration, which is to choose randomly among the
tied features, but also a new tie-break criterion based on the Q-matrix selection criterion of NJ method;

**Split matrix** - Given that the matrix outputted from the imputation cycle is not symmetric, our framework introduce a methodology to deal with this issue. Here, PhyloMissForest explores the possible solutions that can derive from a single matrix, by splitting the matrix into three candidate solutions: lower triangular, upper triangular and mean between both. Moreover, to select the best matrix among the candidate ones, PhyloMissForest proposes the inclusion of phylogenetic criteria such as, LS and ME. With the combination of those techniques, our framework not only ensures that all the content available after the imputation is explored in detail, but also that the matrix chosen is the one that best fits a phylogenetic tree;

**Stop criteria** - The stop criteria of PhyloMissForest also supports the inclusion of phylogenetic LS, ensuring that the algorithm is only concluded, if the actual matrix is worse in a phylogenetic perspective than the matrix of the previous iteration.

Throughout the next subsections, there is a precise description of each strategy proposed in our framework.

### C. Initial Guess

The first methodology included in our proposal is to turn the initial guess of the missing values more guided to our problem. From the properties of a pairwise distance matrix, the matrix outputted from the algorithm needs to be symmetric. Hence, after guessing the initial values for the missing cells, the mean between the distance \( i, j \) and \( j, i \) is calculated and both distances are updated with this average value. With this procedure, we ensure that when the imputation process begins, the matrix with the initial guess respects the properties of a phylogenetic pairwise distance matrix.

### D. Splitting tie-break criteria

During the imputation process of the algorithm, several random forests composed by groups of regression trees are trained. When building a decision tree, in the process of splitting a node into two sub-nodes, sometimes there is a tie between the candidate features. Our framework supports not only the default configuration (random choices), but also a new decision rule based on the Q-matrix of the NJ algorithm.

After guessing the initial values for the missing cells, if the tie-break criteria is based on the Q-matrix decision, the algorithm, using expression 2, obtains the Q-matrix corresponding to the matrix with the initial guess. Once obtained the referred matrix, the algorithm defines a list of priorities for each OTU based on the Q-matrix values.

\[
Q_{ij} = (N - 2)\delta_{ij} - S_i - S_j, \quad \text{where } S_x = \sum_{i=1}^{N} \delta_{xi} \tag{2}
\]

Having the list of priorities defined, when the splitting method is in progress, if a tie in terms of the SSR is observed, the algorithm goes to the list of priorities and chooses, from the Operational Taxonomic Units (OTUs) that are tied, the one that appears first in the list of priorities of the OTU that is being imputed.

### E. Split Matrix

Similarly to the initial guess process, when the algorithm finishes one entire iteration, which is when the imputation loop iterates over all columns, the matrix does not respect the characteristics of a phylogenetic distance matrix. Therefore, we introduce a procedure at the end of the imputation cycle to deal with this problem.

The matrix could simply become symmetric by executing the function developed in the initial guess methodology. However, since now we have a matrix that is resultant of an imputation process, which means that this matrix was not guessed, we could lose important information with this approach. Hence, we developed a new strategy that, from one matrix, analyses three possible solutions. Since the matrix is not symmetric, the lower and upper triangular parts are different. Therefore, if we consider that each one is a possible solution and that the mean between both is also a possible solution, we are considering three candidate matrices. We call this procedure as split matrix.

Considering the three possible solutions, arises the problem of how to choose which matrix is the best. In order accomplish this task, we merged LS to our proposal. Building a phylogenetic tree with NJ, which does not assume the molecular clock hypothesis, implies that sometimes there is a discrepancy between the distances observed in the tree and the distances of the distance matrix. This is the same as saying that not always the phylogenetic tree is additive. For a phylogenetic tree to be additive, the distances in the tree need to correspond with distances in the distance matrix from where the tree was built.

Let \( M \) represent a group of \( N \) pairwise distance matrices. For each pairwise distance matrix, a phylogenetic tree \( T \) will be inferred. Then, the goodness of each phylogenetic tree is measured following the equation (3), where \( M_{ij} \) represents the distance between the OTUs \( i \) and \( j \) in the pairwise distance matrix, while \( T_{ij} \) represents the distance between the OTUs \( i \) and \( j \) in the phylogenetic tree. Since LS chooses the matrix where the discrepancy is more subtle, the matrix choice among \( N \) matrices is the one that minimizes the value of \( S \) in expression (3).

\[
S = \sum_{i,j} (M_{ij} - T_{ij})^2 \tag{3}
\]

In our proposal, aiming to choose the best matrix among the three possible solutions created in the split process, we add to our framework the possibility of using the LS criterion. Merging LS to our proposal enables us to choose which of the three matrices better fits a phylogenetic tree. When all the columns are imputed, the algorithm performs the following steps:

1. Split the matrix into three possible solutions: lower triangular, upper triangular and mean between both;
2. Build a phylogenetic tree for each possible solution using NJ;
3) Recover the distance matrix from each phylogenetic tree;
4) Compute the LS value using the matrix from where the phylogenetic tree is built and the matrix obtained via tree;
5) Choose the matrix that minimizes the value of LS.

Although there is evidence in the state-of-the-art that the ME is a problematic criteria \[12\], we deploy our algorithm in a way where the user is free to choose different decisions throughout the operation. Hence, if the user wants, our framework allows him to choose the best matrix among the three possible solutions using the ME criteria. Lastly, the framework also allows the user to choose the best matrix by only comparing each of the three possible solutions with the matrix of the previous iteration. To accomplish this methodology, the algorithm uses the stop criterion presented in expression (1). Once finished the calculation of the three errors, the chosen matrix is the one that minimizes the value of expression (1). However, if the user chooses to do this, the algorithm will not analyse any phylogenetic tree and, therefore we do not expect an improvement on the performance of the algorithm. Additionally, the user has the possibility to discard the process of analysing the three possible solutions. If this procedure is discarded, the user has the possibility of doing the mean between the lower and upper triangular parts at the end of each column imputation or only when all columns are imputed. However, if the user opts for this option, again any phylogenetic criteria will be included.

F. Stop criteria

Finally, our framework also allows the user to choose between two different ways of calculating the error between the matrix of the previous iteration and the actual matrix. The first consists on applying the original stop criterion presented in Equation (1), while the second one obtains the error by adding the LS criteria.

The basis of the original criterion is to compare position by position between the matrix from the previous iteration and the matrix from the actual iteration. With this methodology, we are just evaluating how close the new matrix is from the matrix of the previous iteration by analysing the numbers of the matrices. Although our framework allows the user to choose this method to obtain the error, in our proposal, we encourage the user to choose the stop criteria based on LS, since it evaluates the phylogenetic trees to attain an improvement in the performance of the algorithm.

As explained above, in our proposal, we use LS to compute the error between the matrix of the previous iteration and the actual matrix. Recalling the definition of LS in phylogenetics, firstly with the pairwise distance matrix a phylogenetic tree is built. Then, a second distance matrix is obtained from the built phylogenetic tree. Having both matrices, the LS value is computed and used as the value of the error. Finished the computation of the new error, the algorithm will now compare the value of the new error with the value of the old error. If the LS value of the actual matrix is lower than the LS value of the matrix from the previous iteration, the algorithm will perform a new iteration with a new imputation loop. Otherwise, if the new error is equal or higher than the error given by the matrix of the previous iteration, it is not necessary to perform another iteration, since the matrix from the previous iteration fits better a phylogenetic tree than the current one.

G. Hyperparameters

Our framework includes six hyperparameters:

- **Bootstrap** - This is a boolean parameter, meaning that the possible values for this parameter are 0 or 1. The main function of this parameter is to enable or not, the bootstrap function;
- **Size of the Bootstrap** - This parameter controls the size of the bootstrapped datasets, meaning that it has a connection with the previous parameter. In other words, this parameter only has influence when the bootstrap is enabled. The possible values for this parameter are floating point numbers between 0 and 1;
- **Max Features** - The aim of this parameter is to define, during the process of building each decision tree, the number of features that are analysed. It only supports positive integer values;
- **Min Leaf** - This parameter defines the minimum number of samples that a node has to contain to be considered as a leaf node. It only supports positive integer values;
- **Max Depth** - The aim of this parameter is to control the maximum depth until each decision tree can grow. If the user wants to limit the growth of the decision trees, the value defined to this parameter should be a positive integer value. Otherwise, the value for this parameter must be -1;
• **Number of trees** - The value of this parameter defines the number of trees of each random forest. It only supports positive integer values.

IV. EXPERIMENTAL EVALUATION

During the experimental evaluation, from a matrix obtained without missing data, we introduce different degrees of missing cells. Then, we apply our methodology to recover the missing information previously introduced. After that, we construct a phylogenetic tree from the original matrix and a phylogenetic tree from the matrix recovered by our methodology. Aiming to evaluate the behaviour of our algorithm, a comparison between those two phylogenetic trees is performed. The distance-based method used to infer the referred phylogenetic trees is NJ. Since this algorithm returns an unrooted phylogenetic tree, in order to compare the tree obtained via imputation with the one constructed from the matrix without missing data, the RF [21] is used as an evaluation criteria.

Given two unrooted trees, A and B, the RF calculates the number of split nodes present in tree A that are not in tree B and vice versa. The value of RF is the sum of both values. In order to get the percentage of the difference between tree A and B, the normalised NRF is used. This metric is obtained by dividing the RF by the maximum possible number of splits, in other words, the maximum possible RF. The maximum possible RF is $2n - 6$, where $n$ is the number of OTUs. The equation for the NRF is present in (4).

$$NRF = \frac{RF}{2n - 6}$$  \hspace{1cm} (4)

Multiplying the value of the NRF by 100, we obtain the score in percentage, that is, the NRF in percentage represents the percentage of error that the imputation introduces. If the NRF is 0%, the reconstruction of the phylogenetic tree is perfect. Note that the Robinson Foulds Distance only inspects the topology of the tree, meaning that, if the bifurcations of the trees are equal between two trees, the NRF will be 0% even if the size of the branches are different.

During the experimental evaluation of our framework, we use three different real-world datasets. 1) A dataset with 9 OTUs of a virus (baculovirus); 2) A dataset called M37x29352, which is an amino acid dataset with 37 OTUs of a fungi(*xylona heveae*), that can be found in rubber trees; 3) A DNA dataset with 55 OTUs of green plants;

A. **Framework study**

During the experimental tests of our thesis, we started by evaluating each proposed decision in our framework, in order to see which are the versions that better behave in a number of real-world datasets. To be possible to evaluate the wide range of possibilities that our framework offers to the user, we divided the study in two main phases:

**Component Evaluation Phase:** In this phase, we tested the split matrix with the three possible criteria to choose the best solution: LS, ME or the same expression as the stop criteria present in (1). We also test two simpler versions: turning the matrix symmetric after each column imputation and turning it symmetric only at the end of all columns imputation. With this, we test five different ways to choose the best matrix, each one combined with stochastic decisions or Q-matrix based decisions in the process of building each decision tree. In terms of the stop criteria we will leave this to change in the second phase, therefore in this phase the stop criteria remains the original one;

**Stop Criterion Phase:** In this phase, based on the results of the previous one, we evaluate the two stop criteria that our framework supports in the combinations of decisions that performed better in the first phase. For both the first and second phase, the initial guess of the missing values is performed with our methodology: first, impute the average value of each column and then turn the matrix symmetric.

We concluded that significant improvements in the performance and accuracy of the inference can be achieved when we incorporate the knowledge provided by phylogenetic LS in the decisions of the algorithm. In this way, the algorithm is able to choose the matrices that best fit phylogenetic quality criteria. Among the combinations of decisions that obtained the best results, it can be highlighted the version that analyses the three matrices and the selection of the best matrix using LS. Moreover, this version solves the ties during decision trees building, by using stochastic decisions and LS as the preferred stop criterion.

B. **Hyperparameter Study**

In every ML method, testing and defining hyperparameters, which is also called as tuning hyperparameters, is one of the most challenging tasks to perform. The most commonly used technique is grid search, in which the user defines a set of values for each parameter and then an exhaustive test of stochastic combinations is tested. This approach can incur in a large number of runs, turning it inefficient. Aiming to turn this task more methodological, a DOE is applied. In [22], there is evidence that the process of tuning hyperparameters in ML is improved by applying DOE. Moreover, the authors explain the method by applying it to a random forest case-study. Hence, we followed part of their method in order to tune our parameters.

Inspecting the values supported for each parameter, we can see that there are parameters that are defined based on the size of the dataset, while there are other parameters that are not defined by connecting their values with the dataset in usage. Since during the experimental evaluation of our algorithm we use three datasets with different sizes, and we also want to fix a combination of parameters that fits all datasets, for the parameters that are not yet defined in function of the size of the dataset, min leaf and max depth, we define them taking it into account. When we say that we are defining a parameter in function of the size of the dataset, it means that the value of the parameter is going to be a floating point number, between 0 and 1, that is multiplied by the size of the dataset to define the value of the parameter. The expression used to calculate the presented process is represented in (5), where the $dim$
represents the dimension of the dataset and the \( \text{perc} \) represents the percentage defined for the parameter.

\[
\text{Parameter} = \text{int}(\text{dim} \times \text{perc})
\]

The hyperparameter study is divided in two cases: bootstrap = 0, which we call non bootstrap case and bootstrap = 1, which we call bootstrap case. For each of the two versions presented above, a study composed of three phases is performed: 1) Study parameter by parameter, aiming to understand the reasonable ranges for each one; 2) Factorial DOE to filter which are the three parameters that have the strongest statistical meaning; 3) Box-Behnken design to define the values of the parameters that remain to define from the previous design.

This study is performed using a software powered by Tibco called Statistica. From this study, we conclude that the optimal combination of parameters for the datasets in usage is the one defined in Table I.

### C. 9x9 Dataset

Starting with the smallest dataset, Figure 3 depicts, for each percentage of missing data tested, the NRF results obtained by our approach and the two ML state-of-the-art approaches. Recall that our main goal is to minimize the value of NRF.

Beginning with the comparison between our approach and the autoencoder, it can be observed that our approach outperforms the autoencoder, since in every percentage of missing data, we achieved a lower value of the mean of the NRF. Moreover, if we want to compare each of our combinations with the same method, the non bootstrap combination recovers better the phylogenetic tree than the autoencoder in every percentage of missing data. In terms of the bootstrap case, we can see that this method is also better than the autoencoder in eleven out of twelve different percentages. The only percentage where autoencoder is slightly better than bootstrap case is in 35% of missing data. Nevertheless, we get 100% of wins against autoencoder since our approach supports the two combinations of parameters. Even so, if we want to compare each one separately, in the non bootstrap case we get 100% of wins, whereas in the bootstrap one, we achieved eleven out of twelve, which gives 91.7%.

Regarding the comparison with the matrix factorization, the performance of our approaches is also better than the matrix factorization. We can see that our approaches win in every dataset with the exception of the ones with 50% of missing data, meaning that, our methodology outperforms matrix factorization in 91.7%.

Apart from the comparison described above, another interesting aspect is the comparison between our two combinations of parameters. As Figure 3 shows, the bootstrap configuration attains the best results in nine out of twelve percentages of missing data tested. The minimum, maximum and standard deviation obtained in each percentage of missing data is also an important information to analyse. As we can see in the box plots of Figure 3, our framework reaches 0% of NRF, which is our main goal, in at least one matrix in the percentages between 5% and 20%, while the other methods only manage to do this in the matrices with 5% and 10% of missing data.

### D. 37x37 Dataset

Similarly to the 9x9 dataset, Figure 4 introduces the results obtained with the dataset with 37 OTUs. Again, we tested percentages of missing data between 5% and 60%, with increments of 5%. For each percentage of missing data, 10 matrices per percentage are tested. During the entire analysis, recall that our main goal is to minimize the value of NRF.

Considering the two configuration of parameters as a single approach, when comparing it with the autoencoder model, from Figure 4 we conclude that we get a lower value of NRF than the referred state-of-the-art method in every percentage of missing data. With this, we get a total of 100% of wins. Moreover, if we compare our two combinations separately with the autoencoder, we conclude that the bootstrap achieves better results in 100% of the cases. The non bootstrap configuration, wins 91.7%, since in the datasets with 10% of missing data, the autoencoder wins by a tight margin (0.5%).

When comparing with matrix factorization, we can observe comparable performance as our approach wins six out of twelve times, which gives a total of 50% of wins. If we analyse our combinations separately, for the non bootstrap case, we get five wins out of twelve, which gives us 41.7% of wins. In the non bootstrap case, we get 50% of wins. Recall that this dataset is an amino acid dataset, which turns the problem more challenging due to the different number of possible characters in the alignment, when comparing with a DNA dataset.

When comparing the non bootstrap combination versus the bootstrap one, it turns out that the bootstrap case wins ten times out of twelve. Therefore, we get a total of 83.3% of wins with the bootstrap version, when comparing with the non bootstrap one.

### E. 55x55 Dataset

Similarly to the 9x9 and 37x37 datasets, we present in Figure 5 the results obtained with the dataset with 55 OTU.

Starting by comparing our approach with the autoencoder, we can see that we clearly outperform the autoencoder in all the twelve percentages of missing data. Moreover, we not only win when considering our two combinations as a single one, but also when comparing them separately, we get 100% of wins in both cases. Note that in this particular dataset, in the all the percentages of missing data, the difference between
the average NRF is never less than 6.8%, thus representing a huge difference in performance between our approach and the state-of-the-art autoencoder.

When we compare our approach with the matrix factorization, it can be observed that we win in every percentage of missing data, which gives us a total of 100% of wins. Additionally, if we compare our two combinations separately with the matrix factorization, we get 100% wins with both versions of our algorithm. Analysing the bootstrap case versus the non bootstrap one, we get a total of 100% of wins with the bootstrap one.

Inspecting the minimum values of NRF, we can conclude that only our approaches are able to recover 0% of NRF in at least one matrix. Additionally, the maximum value obtained in each percentage of missing data is always lower in our approaches than in the state-of-the-art techniques.
F. Phylogenetic trees reconstruction

Given the results presented before and in order to see the capability to recover a phylogenetic tree of our framework, in comparison to the autoencoder and matrix factorization, we depict in Figure 6 an example of a tree recovered with each method comparing with the original tree. This example is from the dataset with 9 OTUs with 5% of missing cells.

From the analysis of Figure 6, it can be verified that, for both autoencoder and matrix factorization, the recovered phylogenetic trees have differences from a topological perspective. We can identify the presence of two species groups: 1) XcGV, CpGV and PxGV; 2) AcMNPV, OpMNPV and BmNPV. They not only appear in the tree obtained with full information, but also in the trees obtained with the two referred methods. However, the relationships of this groups with the other OTUs were lost in the imputation process. Given that, for both state-of-the-art algorithms, the trees are recovered with 16.6% of NRF.

On the other hand, PhyloMissForest is able to fully recover the topology of the original tree. Although our metric of evaluation not only evaluates topology, even in the branch lengths, our methodology obtains a tree almost equal when comparing with the tree obtained with the full matrix. Note that we only present one tree, since for both bootstrap and non bootstrap cases, the tree is fully with 0% of NRF.

V. Conclusions

This Thesis has been focused on defining a ML-based framework to allow the reconstruction of phylogenetic trees from distance matrices with missing data. To accomplish this task, we took as a reference a method called MissForest, which is a random forest based imputation algorithm. Aiming to guide it for our problem, we devised a number of search strategies and techniques, coupled with phylogenetic knowledge and criteria, to efficiently tackle the missing data problem. When using our framework, which we called PhyloMissForest, the user is able to choose different decisions throughout the algorithm. Firstly, our initial guess of the missing values is performed using a methodology that respects the properties of a phylogenetic distance matrix. Secondly, we introduce a strategy in which, from the matrix resultant of the imputation process, creates three candidate matrices. Aiming to choose the best matrix among the possible solutions, our framework supports the inclusion of LS, ME or the baseline criteria. Moreover, if the user opts to discard this methodology, our framework has the possibility of turning the matrix symmetric during or at the end of the imputation cycle. Finally, our framework supports two different stop criteria, baseline and LS.

During the experimental tests of our Thesis, we started by evaluating each proposed decision in our framework, in order to see which are the versions that better behave in a number of real-world datasets. We concluded that significant improvements in the performance and accuracy of the inference can be achieved when we incorporate the knowledge provided by phylogenetic LS in the decisions of the algorithm. In this way, the algorithm is able to choose the matrices that best fit phylogenetic quality criteria. Among the combinations of decisions that obtained the best results, it can be highlighted the version that analyses the three matrices and the selection of the best matrix using LS. Moreover, this version solves the ties during decision trees building, by using stochastic decisions and LS as the preferred stop criterion.

We used a DOE to tune the parameters of our algorithm. By using this method, we are avoiding the well known exhaustive hyperparameter tests. From the insights provided by DOE, we proposed two combinations of parameters: non bootstrap and bootstrap. Finished the process of studying the framework and tuning the hyperparameters, we compared our two configurations with the current state-of-the-art competitors: matrix factorization and autoencoder. We conducted all the experimental study in three real-world datasets with sizes between 9 and 55 OTUs and missing data percentages between 5% and 60%. In an overall perspective, in the DNA datasets, our framework outperforms the state-of-the-art methods in all the percentages of missing data tested. In the amino acid dataset, our framework wins in all the cases when comparing with the autoencoder, whereas in the comparison with matrix factorization, our framework is able to reach comparable results (50% of wins).

In conclusion, PhyloMissForest fills the gaps identified in the state-of-the-art methods, since we not only do not allow missing entries to be filled with negative values, but also propose a framework that supports the inclusion of phylogenetic criteria, in the decisions of the algorithm. Given the results herein presented, we believe that our work represents a valuable tool to improve phylogenetic reconstructions in the presence of missing data.

VI. Future Work

Throughout the experimental results, we saw that our framework is a valuable tool for the phylogenetic research. Nevertheless, it opens possible improvements in the future.

The random forest technique is widely used in ML approaches. One of the key problems of this approach is that these models are computationally demanding when it comes to training them. Therefore, efficient parallel approaches and implementations on GPU were developed to turn the training process of this model faster [23]. By applying the referred techniques, we expect an improvement in the process of training the random forest and therefore, a boost on the computational performance of our proposal.

Despite the good results obtained, there is always room for introducing ideas that can improve our main goal, which is to minimize the NRF. Although it is obvious, our results confirmed that there is a tendency for an increase in the NRF, with the increase of the percentage of missing data. A promising approach lies in the hybridization of our current method with other alternative methodologies that operate directly at the alignment level, which can play an important role to boost the accuracy of the inference [17]. By combining both methodologies, a first filter of the missing data would be
applied and then, when our current approach is executed, the matrix would have less missing data.

REFERENCES


