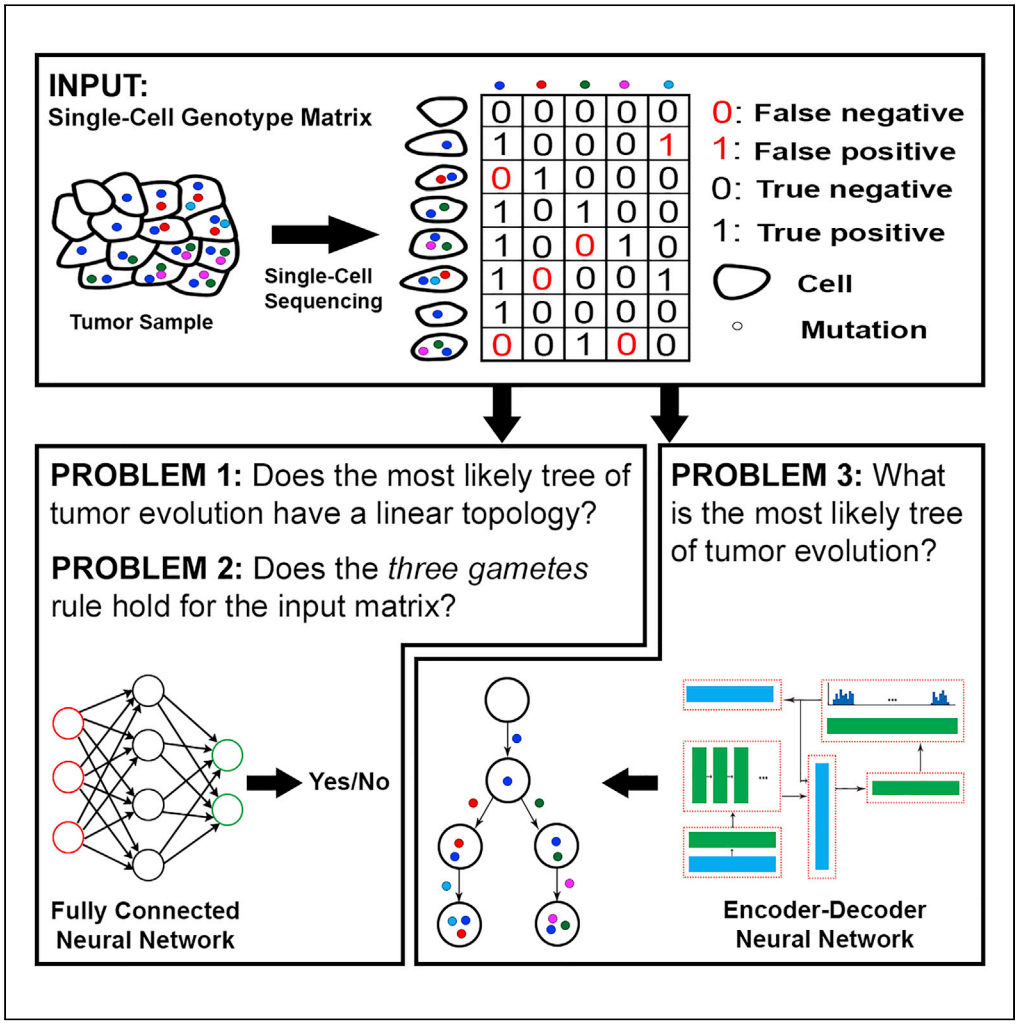
iScience

CellPress
OPEN ACCESS

Article

Tumor Phylogeny Topology Inference via Deep Learning



Erfan Sadeqi Azer, Mohammad Haghir Ebrahimabadi, Salem Malikić, Roni Khardon, S. Cenk Sahinalp

cenk.sahinalp@nih.gov

HIGHLIGHTS

We describe the first application of deep learning in studying tumor evolution

A fast method for identifying the existence of divergent tumor subclones is presented

We present a neural networks-based solution for the three-gametes rule testing

Reinforcement learning can be utilized for inferring complete tumor phylogenies.

Sadeqi Azer et al., iScience 23, 101655
November 20, 2020
https://doi.org/10.1016/j.isci.2020.101655

iScience

CellPress
OPEN ACCESS

Article

Tumor Phylogeny Topology Inference via Deep Learning

Erfan Sadeqi Azer,[1,3,4] Mohammad Haghir Ebrahimabadi,[1,2,3] Salem Malikić,[1,2] Roni Khardon,[1] and S. Cenk Sahinalp[2,5,*]

SUMMARY

Principled computational approaches for tumor phylogeny reconstruction via single-cell sequencing typically aim to build the most likely perfect phylogeny tree from the noisy genotype matrix – which represents genotype calls of single cells. This problem is NP-hard, and as a result, existing approaches aim to solve relatively small instances of it through combinatorial optimization techniques or Bayesian inference. As expected, even when the goal is to infer basic topological features of the tumor phylogeny, rather than reconstructing the topology entirely, these approaches could be prohibitively slow. In this paper, we introduce fast deep learning solutions to the problems of inferring whether the most likely tree has a linear (chain) or branching topology and whether a perfect phylogeny is feasible from a given genotype matrix. We also present a reinforcement learning approach for reconstructing the most likely tumor phylogeny. This preliminary work demonstrates that data-driven approaches can reconstruct key features of tumor evolution.

INTRODUCTION

Cancer is an evolutionary disease characterized by progressive accumulation of somatic mutations in tumor cells. Deciphering the evolutionary history of a given tumor represents an important problem in studies of cancer and can help us in better understanding of several clinically important aspects of the tumor, including progression, metastatic spread, the existence of divergent subclones evolving on different branches of the tumor phylogenetic tree, and many others.

Due to the importance of the problem, there have been rapid developments in the design of principled computational methods for tumor phylogeny inference. Many of these methods use bulk sequencing data where DNA from millions of cancerous and normal cells are sequenced together. Tree inference from this type of data is typically based on the use of cancer cell fraction of detected variants – in particular, single-nucleotide variants (Strino et al., 2013; Deshwar et al., 2015; El-Kebir et al., 2015, 2016; Malikic et al., 2015; Popic et al., 2015; Donmez et al., 2017; Satas and Raphael, 2017; Myers et al., 2019; Husić et al., 2019), but also copy number (e.g. (Zaccaria et al., 2017)) and structural variants (e.g. (Eaton et al., 2018; Ricketts et al., 2019)). While being cost-effective, the low resolution of bulk sequencing data is a limiting factor in tumor evolution modeling. In particular, bulk sequencing data from a single tumor sample typically admit a linear topology as an optimal solution under common tree-scoring models (Donmez et al., 2017; Gerstung et al., 2020). However, inferring whether the underlying tumor includes divergent subclones, which evolve through distinct branches of the tumor phylogeny, represents an important step toward better understanding tumor progression and improving treatment design.

Recent technological developments have enabled researchers to perform single-cell sequencing (SCS) experiments, where DNA from an individual cell is extracted, amplified, and sequenced. SCS provides high-resolution data for studying tumor evolution at unprecedented detail, e.g., it offers the possibility to identify branching topologies with high confidence or to solve the general problem of inferring the complete history of tumor evolution, even when all of the sequenced single cells are extracted from a single-tumor biopsy sample. Early developments in inferring tumor evolutionary history from SCS data include probabilistic methods such as SCITE (Jahn et al., 2016), OncoNEM (Ross and Markowetz, 2016), which employ the "infinite sites" assumption, and SiFit (Zafar et al., 2017), which relaxes this assumption by allowing violations

[1]Department of Computer Science, Indiana University, Bloomington, IN 47408, USA

[2]Cancer Data Science Laboratory, Center for Cancer Research, National Cancer Institute, National Institutes of Health, Bethesda, MD 20892, USA

[3]These authors contributed equally

[4]Present address: Google LLC

[5]Lead Contact

*Correspondence: cenk.sahinalp@nih.gov

https://doi.org/10.1016/j.isci.2020.101655

at a given "cost". More recently, SPhyR (El-Kebir, 2018), a combinatorial optimization approach based on "Dollo" parsimony, and SiCloneFit (Zafar et al., 2019), which is an improved version of SiFit, were proposed. Finally, PhISCS-BnB (Sadeqi Azer et al., 2020), another combinatorially optimal tool utilizing a branch-and-bound strategy, and ScisTree (Wu, 2020), a neighbor joining–based heuristic, are among the most recently developed methods that offer significant improvements in running time.

When both single-cell and bulk sequencing data of a tumor sample are available, two of the latest methods, namely B-SCITE (Malikic et al., 2019a) and PhISCS (Malikic et al., 2019b), can provide more accurate tumor phylogenies.

B-SCITE is a probabilistic method that integrates SCITE (Jahn et al., 2016) and CITUP (Malikic et al., 2015), whereas PhISCS is based on the use of combinatorial optimization and has two distinct implementations: PhISCS-I uses integer linear programming while PhISCS-B formulates the same problem as a boolean constraint satisfaction program and solves it via available solvers for weighted max-SAT. Both approaches can also be applied to data sets consisting solely of SCS data.

As summarized above, available methods for tumor phylogeny reconstruction by the use of SCS data have important limitations. First, many of these methods employ Infinite Site Assumption, ISA, (even though some offer a provision for limited loss and concordant gain of mutations) and assume a uniform noise level (false negative as well as a false positive rate) – both subject to change with advances in our understanding of tumor evolution and SCS technology. More importantly, these methods aim to infer the most likely tumor phylogeny and for that eliminate noise (due to, e.g., allele dropout or low sequence coverage) with a maximum likelihood/parsimony approach. As such, they all aim to solve an NP-hard problem from scratch and thus cannot scale up to handle large SCS data sets. Even when the goal is to infer basic topological features of the tumor phylogeny rather than reconstructing it entirely, these methods cannot easily handle SCS data involving a few hundred mutations and cells. As a result, fast techniques for inferring key features of a tumor phylogeny, e.g., those that can discriminate linear from branching topologies, especially for SCS data sets with high noise levels (as per the current practice) are in high demand. Similarly, it is desirable to quickly infer whether any noise elimination is necessary for constructing a perfect phylogeny. Finally, each of the existing tools has required a great deal of human effort in algorithmic design and implementation, as each technological advance in data generation necessitated the development of completely novel methodologies. It is thus highly desirable to have a general computational approach that can adapt to technological change, simply through training it with new data, without the need for explicit objective or noise profile modeling.

It may be possible to address these limitations via a "data-driven", machine learning approach that considers a general set of functions and choose one that best fits a training data set – which can be simulated or obtained through real-world measurements. Such an approach can not only reduce the inaccuracies in noise profile modeling but also identify implicit underlying patterns in the data or problem toward developing more realistic objectives. Recent advances in deep learning (DL) have demonstrated remarkable generalization of formulations for solving many problems (e.g., mastery over games such as Chess, Go (Silver et al., 2017), and Poker or handling natural language tasks via BERT (Devlin et al., 2019) and most recently RoBERTa (Liu et al., 2019)), and it is possible that a single DL architecture may succeed in inferring distinct properties of tumor phylogenies when properly trained on a sufficient number of data sets.

In recent years, many computing applications have experienced a shift to data-driven approaches, from deciphering handwritten text (e.g., (Ciregan et al., 2012) for digit recognition) to natural language processing, e.g (Devlin et al., 2019; Liu et al., 2019). Problems with poorly understood/formulated objectives such as those in structural biology (e.g., *AlphaFold* (Senior et al., 2020) for inferring the 3D structure of a protein sequence) seem to benefit considerably from DL methods. However, we are not aware of any data-driven approach for tumor phylogeny inference.

In this work, we offer the first data-driven tumor phylogeny reconstruction methods to address the limitations of existing strategies. We have used SCS data with deep neural networks and reinforcement learning to infer topological features of a tumor phylogeny, as well as the most likely evolutionary history of a tumor. In order to achieve this, we had to overcome several distinct challenges: (1) The neural network should ideally be designed so as to handle a varying number of cells and mutations. Alternatively, for models

with fixed-sized inputs, it is desirable to use our domain knowledge to prepare the data in a manner that will facilitate success in predictions. (2) Given the use of neural networks, one needs a large number of samples for proper training. Unfortunately, the number of publicly available tumor SCS data sets are not sufficiently large to train DL models; thus, we needed to produce a large number of simulated SCS data sets. (3) Errors/noise in SCS data add further complexity to the problem, and the proposed DL framework had to be evaluated in terms of noise tolerance. (4) The chosen architecture also expects a specific type of supervision which we must be able to compute. In order to perform noise reduction/elimination in the input "genotype matrix" (see below for a formal definition) extracted from SCS data, it is possible to offer supervision in the form of a data set of noisy inputs along with their denoised versions. An alternative and cheaper supervision is offered by a feedback mechanism for determining whether a candidate output of the neural network is successfully denoised. A third alternative is given by a cost function that indirectly helps supervise a reinforcement learning process.

Inspired by novel DL approaches to combinatorial problems such as the "reinforce policy gradient algorithm" (Williams, 1992) for the traveling salesperson and the knapsack problems (Bello et al., 2017) and, to a limited degree, the "NeuroSAT" approach (Selsam et al., 2019) for the satisfiability problem using a single-bit supervision, we established a computational framework to address all the above challenges as follows successfully. (i) We employed a reinforcement learning approach to train a denoising model without a need for the ground truth. The cost function we used is novel and problem specific (see Section S1.3). (ii) We devised a novel method to transform the input matrix (obtained from noisy SCS data) to feed to the neural network. This method incorporates the noise rates as well as coordinates and values of entries in the given matrix (see Section S1.3.1). This representation also supports training and testing instances of different sizes and encourages robustness to permutation of rows and columns. (iii) We also present an example for incorporating domain knowledge to improve the performance of a machine learning model: this was achieved by sorting the columns of the input genotype matrix in a preprocessing step (see Section S1.2). (iv) The simulated input data we use with a given number of mutations and cells were obtained through a pipeline we developed – which along with the simulation tools from previous work, allows us to construct unbiased data sets to train our models.

Results

We Study Three Major Problems

First, given a noisy SCS data set in the form of a genotype matrix, we consider the problem of inferring whether the most likely tumor phylogeny has linear topology or contains at least one branching event. The method we devised for this problem succeeds to learn and differentiate the two topology types in ~98% of randomly chosen noisy genotype matrices of size 100×100. The trained model runs at least 1000 times faster than the fastest available baseline techniques (see Table 2), which needs to first denoise the input and then examine the tree. The trained model is noise tolerant, scalable, and can analyze real data sets (see Section 3.2). Moreover, having an efficient prediction function allows us to systematically study the relationship between the amount and type of noise in the input and the information preserved in the data to enable correct prediction.

Second, we present results on the problem of inferring whether a given binary matrix is conflict free and thus admits a perfect phylogeny or does not satisfy the so-called three-gametes rule. Our DL approach for this binary classification problem offers notably more accurate results in comparison to support vector machines. Crucially, sorting the columns of the input genotype matrix according to their binary representation as a preprocessing step further improves the accuracy of the trained models by 13–27% (see Table 3).

Third, we study the general problem of reconstructing the complete most likely tumor phylogeny. For solving this problem, we introduce a reinforcement learning-based approach, which finds a solution for, e.g., 92% of the randomly generated noisy genotype matrices of size 10×10. Furthermore, our trained model on noisy genotype matrices of size 10×10 successfully solves a notable proportion of larger noisy genotype matrices, demonstrating the generalizability of our approach. See Table 1 for a summary of our approach's key characteristics.

PRELIMINARIES

Given a positive integer z, let $[z]$ denote the set $\{1,2,\ldots,z\}$. For a given matrix M, we let $|M|_0$ denote the number of nonzero entries in M. For any boolean proposition P, we use Iverson bracket notation as follows:

Problems	The Key Characteristics of Our Solutions
Branching Inference	• Based on a two-layer fully connected neural network • Scalable • Noise tolerant • Faster than available alternatives
Noise Inference	• Based on a two-layer fully connected neural network • Has better accuracy compared to the support vector machines • Its accuracy improves with our input preprocessing • Provides insights about the capacity of machine learning models in studying key features of tumor evolution • Does not provide improvement in running time over the available sophisticated combinatorial solution
Noise Elimination	• Based on a reinforcement learning framework • Generalizes across matrix sizes, i.e., models trained on small matrices can solve larger matrices • Typically slower than the existing methods

Table 1. A Summary of the Key Characteristics of Our Solutions for the Problems Defined in "Preliminaries"

$$[P] = \begin{cases} 1 & \text{if } P \text{ is true} \\ 0 & \text{otherwise.} \end{cases} \quad \text{(Equation 1)}$$

Assume that we have performed an SCS experiment where n single cells, denoted by $C_1, C_2, \ldots, C_n$, were sequenced and m putative mutations, denoted by $M_1, M_2, \ldots, M_m$, were reported (in this work, we focus on single-nucleotide variants at loci that are homozygous in normal cells). We define a "genotype" of cell C_i as a binary vector of length m having j-th coordinate equal to 1 if and only if C_i harbors mutation M_j. We distinguish between the true and noisy genotypes of a cell, where the latter is obtained from SCS data and typically contains some false mutation calls.

We can combine true genotypes of the sequenced cells into a binary matrix A having n rows (corresponding to the cells) and m columns (corresponding to the mutations). More formally, $A[i,j]=1$ if and only if cell C_i harbors mutation M_j. Similarly, we define matrix A' by combining noisy genotypes. In other words, $A'[i,j]=1$ if and only if, after single-cell data processing and mutation calling steps, mutation M_j was reported to be present in cell C_i. For the convenience of notation, below, we will mostly use $a_{i,j}$ and $a'_{i,j}$ assuming that they are equal to $A[i,j]$ and $A'[i,j]$, respectively.

Single-cell genotype matrix D (of dimension $n \times m$ and with terms denoted by $d_{i,j}$) is said to represent a "perfect phylogeny" if it satisfies the "three-gametes" rule, i.e.,

$$\forall i,j,k \in [n] \wedge \forall g,h \in [m], \begin{vmatrix} d_{i,g} & d_{i,h} \\ d_{j,g} & d_{j,h} \\ d_{k,g} & d_{k,h} \end{vmatrix} \neq \begin{vmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{vmatrix}.$$

Notice that the rows and columns in the definition above can be in any order. In the case of equality, the row triplet (i,j,k) and column pair (g,h) are said to have a "conflict" or a "three-gametes rule violation". We also refer to a matrix without such conflict as a "conflict-free matrix". It is well known that, under the infinite sites assumption, for any conflict-free matrix, there exists a phylogenetic tree depicting the evolutionary history of the tumor such that (i) the set of mutations assigned to the edges of the tree equals the set of mutations given by the matrix, (ii) each mutation appears exactly once in the tree, (iii) each of the single cells represents exactly one leaf of the tree, and (iv) the set of mutations occurring on the path from the root to a given single cell (i.e., leaf) equals to the set of mutations present in the cell (this set is given by the related row of the matrix) (Gusfield, 1997; Malikic et al., 2019b; Edrisi et al., 2019). In addition, it is trivial to show that, under the infinite sites assumption, any noise-free single-cell genotype matrix is conflict free. Therefore, instead of searching for the most likely tree of tumor evolution directly in the space of trees (as was

done, for example, in (Jahn et al., 2016) and (Malikic et al., 2019a)), we can first search in the space of conflict-free matrices (as was done, for example, in (Malikic et al., 2019b)) and then, using the algorithm described in (Gusfield, 1997), convert the obtained matrix into a tree. Note that in this work, we also consider "clonal trees" of tumor evolution, typically obtained from phylogenetic trees by discarding the attachment of single cells as leaves. We refer to the section "Tree models of tumour evolution" in (Malikic et al., 2019a) for a detailed definition of the clonal tree.

To generate conflict-free matrices representing instances of A, we used the most recent version of the *ms* simulator, which was first introduced in (Hudson, 2002) and has been used in earlier studies on phylogeny inference from bulk or single-cell sequencing data (Bonizzoni et al., 2018; El-Kebir, 2018). To obtain instances of A' which are not conflict free, we added noise to A by assuming an independent and identically distributed (i.i.d.) noise process across the mutated loci (similarly as was done in most of the available studies, e.g., (Jahn et al., 2016, Ross and Markowetz, 2016, Malikic et al., 2019a, b)). More precisely, we choose to flip each entry independently with a probability depending on the value of the entry and predefined false positive (α) and false negative (β) rates. To ensure the correctness of the labels and get a fixed number of instances that are not conflict free, we check each generated matrix A' to verify that it contains at least one conflict. If it does not, we repeat the above procedure until a matrix A' containing one or more conflicts is obtained. See Section S3 for details related to the use of *ms* and generation of matrices A'.

Under the assumption of independent and identically distributed (i.i.d.) noise process as mentioned above, the conditional probability that a conflict-free matrix B (with entries denoted by $b_{i,j}$) generates A' is as follows:

$$P(A'|A = B, \alpha, \beta) = \alpha^{N_{10}} \cdot \beta^{N_{01}} \cdot (1-\alpha)^{N_{00}} \cdot (1-\beta)^{N_{11}} \quad \text{(Equation 2)}$$

where N_{pq} for each pair $(p,q) \in \{(0,0),(0,1),(1,0),(1,1)\}$ is defined as

$$N_{pq} = \sum_{(i,j) \in [n] \times [m]} [a\prime_{i,j} = p] \wedge [b_{i,j} = q]. \quad \text{(Equation 3)}$$

The detailed derivation of the above formula can be found in Equations (2) and (3) in (Malikic et al., 2019b). Note that $P(A'|A=B,\alpha,\beta)$ represents the likelihood that B is the true single-cell genotype matrix, assuming that noisy genotype matrix A' was obtained from SCS data (or, shortly, the "likelihood of B", as will be used throughout the rest of the manuscript).

In the remainder of this section, we will provide a formal definition and brief overview of each of the three problems considered in this work. Then, in Sections S1.1, S1.2, and S1.3, solutions to each of these problems are discussed in more detail.

Branching Inference Problem

As mentioned in "Introduction", the problem of inferring whether a tumor contains divergent subclones evolving through distinct branches of its phylogeny has potential applications in improving cancer treatment. This motivated us to consider the following computational problem.

no_branching Problem. Given a noisy binary matrix A' obtained as the output (after mutation calling step) of a single-cell sequencing experiment, decide whether the most likely clonal tree of tumor evolution of the sequenced tumor has a linear topology or a topology which contains at least one branching event.

To solve the above problem, instead of inferring the complete most likely clonal tree of tumor evolution and then checking whether it has linear topology or not, here we obtain equivalent results by working solely in the space of binary matrices. This is possible since the set of conflict-free matrices, which imply linear topology, can be precisely characterized. Before providing the formal mathematical characterization of this set, we introduce the set of "staircase matrices". A staircase matrix is any binary matrix S such that (i) $S[i,j] \leq S[i+1,j]$ for all pairs of integers (i,j) such that $1 \leq i \leq n-1$ and $1 \leq j \leq m$ and (ii) $S[i,j] \leq S[i,j+1]$ for all pairs of integers (i,j) such that $1 \leq i \leq n$ and $1 \leq j \leq m-1$, where n and m denote the number of rows and columns of S, respectively. It was recently shown in (Malikic et al., 2020) that a binary matrix D is conflict free and implies a clonal tree with linear topology if and only if its columns and its rows can be reordered so that the newly obtained matrix belongs to the set of staircase matrices. In other words, the set of conflict-free matrices that imply trees having linear topology consists exactly of all binary matrices that can be converted

to staircase matrices by reordering their rows and columns. In solving the *no_branching* problem, we extensively rely on this fact.

We say that any conflict-free binary matrix D which implies a clonal tree with linear topology has the "no-branching" property. Given the above, *no_branching* problem can also be defined as the problem of inferring whether the most likely denoised version of a given binary matrix A' has no-branching property.

It was very recently shown that the *no_branching* problem is NP-hard even when a false positive rate α of single-cell data is set to 0 (We found one proof of this in the upcoming publication Weber and El-Kebir (2020)).

We study this problem in the machine learning framework introduced as follows: Given a conflict-free matrix D, we define a function $Sing(D):\{0,1\}^{n\times m}\rightarrow\{0,1\}$ which takes value 1 if D has the no-branching property, i.e., implies a linear topology, and 0 otherwise. Our goal for the *no_branching* problem is to choose a function $f(D):\{0,1\}^{n\times m}\rightarrow\{0,1\}$ from a domain of functions described in Section S1.1, such that f approximates $Sing$ with the highest possible accuracy, where the accuracy of f is defined as the proportion of potential input matrices D where $f(D)=Sing(D)$.

Noise Inference Problem

In the *noise_inference* problem, our goal is to investigate whether neural network models can help us to successfully check whether a given binary matrix admits perfect phylogeny, which is equivalent to checking whether the matrix contains at least one conflict. So we consider the following problem.

noise_inference Problem. Given a binary matrix A', decide whether A' is conflict free.

This problem is well studied in the literature and has a linear time algorithm (Gusfield, 1991). Here, we study the capacity of machine learning models to train to solve this problem. We start by imposing the machine learning framework on the *noise_inference* problem as follows: Let the function $Icf(A'):\{0,1\}^{n\times m}\rightarrow\{0,1\}$ have value 1 if A' contains at least one conflict and 0 if it is conflict free. Our goal for the *noise_inference* problem is to choose a function $f(A'):\{0,1\}^{n\times m}\rightarrow\{0,1\}$ from a domain of functions described in Section S1.2, such that f approximates Icf with the highest possible accuracy, where this time, the accuracy of f is defined as the proportion of potential input matrices A' where $f(A')=Icf(A')$.

Noise Elimination Problem

The problem of denoising a binary matrix obtained from SCS data to obtain a conflict-free matrix has recently attracted attention in the field of tumor phylogenetics (Ciccolella et al., 2018; Edrisi et al., 2019; El-Kebir, 2018; Malikic et al., 2019b; Sadeqi Azer et al., 2020). The general denoising problem, which we call the *noise_elimination* problem, is defined as follows:

noise_elimination Problem. Given the input consisting of noisy genotype matrix A' and noise parameters α and β, our objective is to find conflict-free matrix B such that the likelihood of B, i.e., the probability $P(A'|B,\alpha,\beta)$, is (ideally) maximized.

The *noise_elimination* problem is the most computationally challenging problem considered in this paper since it cannot be easily formulated as a classification, regression, or clustering problem, which are better suited for DL approaches. More importantly, the problem is NP-hard (Chen et al., 2006), and available principled solvers deploy sophisticated methods such as mixed-integer linear programming, constraint satisfaction programming, Markov chain Monte Carlo toward its solution for practical instances (Chen et al., 2006, Jahn et al., 2016, Malikic et al., 2019a, b). Finally, since the problem is computationally intractable, it is not possible to construct large matrices with known optimal solutions to be used for training a neural network. This issue alone makes it impossible to use several successful machine learning techniques for our purposes.

EXPERIMENTS

In this section, we discuss our experimental setup and results. Additional details related to hardware specifications, the generation of simulated data, and hyperparameter tuning are presented in Supplementary Sections S2, S3, and S4 to ensure reproducibility.

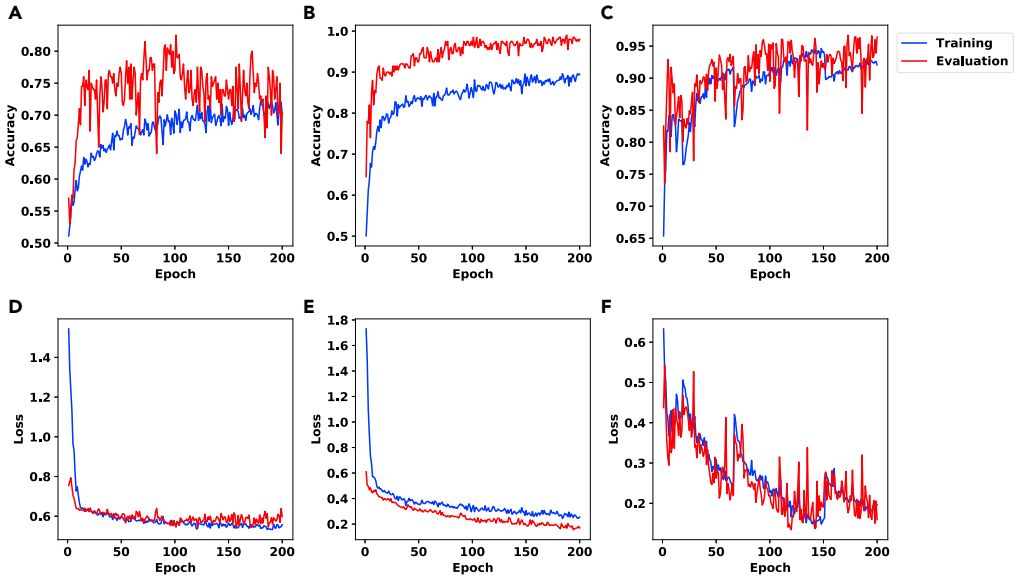


Figure 1. The Accuracy (First Row) and Binary Cross Entropy Loss (Bishop, 2006) (second row) Plots Corresponding to the Results for the *no_branching* Problem and the Architecture Described in Section S1.1

We used a data set of 2000 matrices with dimension 100×100 and hidden layer of size 10 in (A,D), the same data set as in (A,D) with hidden layer of size 100 in (B,E), and a data set of 10,000 matrices with dimension 500×500 and hidden layer of size 100 in (C,F).

Experimental Setup

All experiments presented here were performed using DL nodes in "Carbonate", a computation cluster at Indiana University (Stewart et al., 2017). In order to ensure the reproducibility of our results, we provide the specifications of these nodes, along with the version numbers of the packages used, in the README.md file of our repository.

We note that the accuracy values in Figures 1, 3, and 4 are higher for the evaluation phase than for the training phase. This reflects the fact that the error during training was measured in a noisy manner using dropout, which is necessarily worse than the true error rate of the network.

Branching Inference

Recall the *no_branching* problem, where the goal is to identify whether the tumor phylogeny has a linear or a branching topology.

Learning Curves

In the first set of experiments for this problem, we split the data set to training and evaluation sets with a ratio of 9 to 1. We designed several experiments with different settings and all of them show success in predictions. We include results for three settings in Figure 1: (A,D) a data set of 2000 matrices of size 100×100 and a hidden layer of size 10, (B,E) same data set, now with a hidden layer of size 100, and (C,F) a data set of 10,000 matrices of size 500×500 and hidden layer of size 100. In each of these experiments, we run the training process for 200 epochs. Note that the model distinguishes up to 98% of randomly chosen instances of size 100×100 correctly – see panel (B) in Figure 1.

Comparison of Running Time for the no_branching Problem

We performed an experiment to compare the running time of our approach and recently published method PhISCS (Malikic et al., 2019b), which we use as a baseline for the *no_branching* problem.

In order to assess the performance improvement obtained by our approach, we ran it on two simulated genotype matrices, one 100×100 and another 500×500. Each matrix was then subjected to random noise

Input Matrix Size	100×100	500×500
Time: Our approach	0.0372	0.0537
Time: PhISCS	>80	>3772

Table 2. A Comparison of the Running times (Measured in Seconds) between Our Approach and the Baseline for the *no_branching* Problem. For our approach, we tested 50 noisy cases (α=0.002, β=0.2) for both matrix sizes and report the maximum running time we observed. For the baseline, we report the best-case running time for noise-free input matrices (adding noise increases the running time substantially).

(α=0.002 and β=0.2) 50 times before we applied our approach; we reported the worst running time of our approach out of these 50 runs.

PhISCS first denoises the input matrix before it builds the tree, based on which we can check whether its topology is linear or not. Based on our experience with running PhISCS, its running time depends heavily on the noise levels, i.e., the higher the noise level, the longer its running time. Therefore, we decided to run PhISCS on the noise-free versions of these two input genotype matrices. Despite this disadvantageous setting for our approach, it turned out to be more than 1000 times faster than PhISCS, as can be seen in Table 2. Note that, since the running time of PhISCS was prohibitive for the larger matrices, we could not conduct a more comprehensive study of its output characteristics, including its accuracy.

Evaluation of Noise Tolerance for the no_branching Problem

In another experiment, we tested the noise tolerance of our model on five different values for β/α, i.e., the false negative to false positive rate ratio. For each value in Figure 2, we plotted the percentage of inputs whose topologies are correctly distinguished as a function of the false negative rate β. We observe that when the false positive value is relatively low, e.g., in the case where α=0.03, β=0.3 in Figure 2 panels (C,D,E), the accuracy of the model remains close to perfect across all (realistic) values of the false negative rate β. Notice that as β/α increases in panels (C,D,E), the accuracy increases.

Note that the efficiency of our approach as shown in Table 2 enabled us to run an extensive number of experiments to obtain the data points in Figure 2. The plots in this figure suggest a new understanding of the relationship between the amount of preserved information that can be captured by our neural network in the presence of different noise levels. For example, for settings where false negative and false positive rates are equal, a false negative rate of $\leq$0.15 preserves the deduced topological outlook fairly consistently. The drop in accuracy after this point seems steeper.

Noise Inference

We trained and evaluated our proposed neural network for the *noise_inference* problem introduced in Section S1.2 on multiple data sets. We considered two sizes for the input matrices: 10×10 and 25×25. For each size, we experimented with two distinct data sets that differed with respect to the amount of noise introduced. Note that in the context of this problem, unlike both the *no_branching* problem and the *noise_elimination* problem, the lower the noise level, the harder the decision problem becomes. In order to assess the full capabilities of our model, we used lower values for α and β than typical false positive and false negative rates observed in real SCS data sets. (As mentioned in "Preliminaries", even at lower noise levels, we have ensured that the noisy matrices include at least one conflict.)

In Figure 3, panels (A,C) show accuracy and binary cross entropy loss (Bishop, 2006) for training and evaluation data sets, respectively, comprising 2*M* and 2*K* matrices, each of size 10×10, where half of each set is noisy with α=0.002 and β=0.1 (with the remaining half being conflict free). Panels (B,D) show results for $\alpha=4\times10^{-4}$ and β=0.02 (again accompanied by an equal number of conflict-free matrices, both in training and evaluation). We observe an accuracy of 90% in the data set with a higher noise level. This confirms that distinguishing matrices that contain at least a conflict from those without any conflict gets harder as the noise level decreases. It is worth mentioning that the noise levels observed in practice are higher than these values.

Results for matrices of size 25×25 are shown in Figure 4. We used $\alpha=3.2\times10^{-4}$ and β=0.016 in panels (A,C) and $\alpha=4\times10^{-4}$ and β=0.02 in panels (B,D).

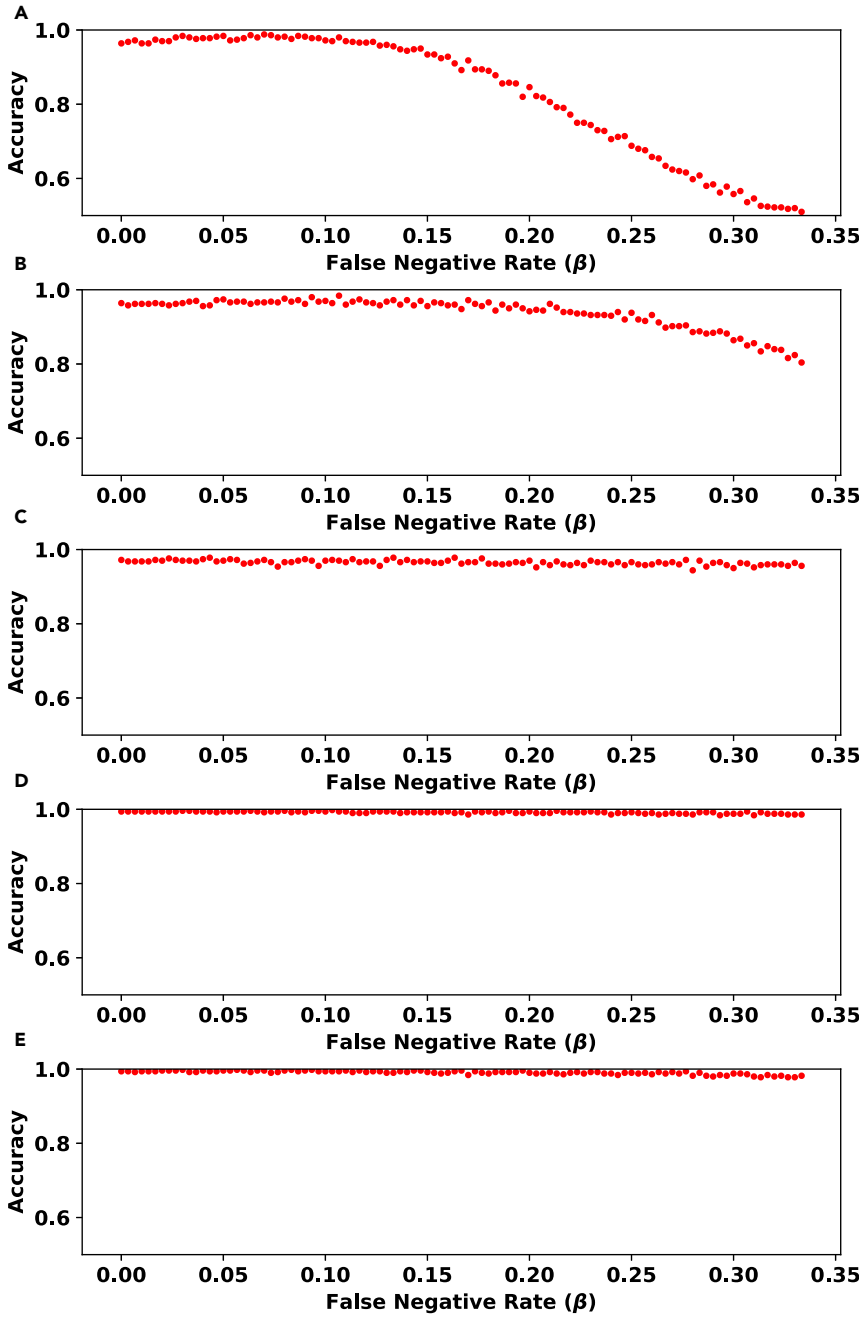


Figure 2. The Evaluation of Noise Tolerance for Our Model for the *no_branching* Problem

In each panel, the y axis (accuracy) represents the fraction of instances classified correctly from 500 randomly chosen instances. These instances are not seen during training and consist of an equal number of instances from each topology type (see Section S1.1). The trained model consists of two layers with size 100 and 0.9 dropout rate. This model was trained using 2000 matrices of size 100×100 with no noise for 200 epochs.

The five panels differ in the ratio of false negative to false positive rates: (A) $\alpha=\beta$, (B) $\alpha=\beta/2$, (C) $\alpha=\beta/10$, (D) $\alpha=\beta/100$, and (E) $\alpha=\beta/1000$. Note that the noise tolerance of the method gets more robust as the relative value of α decreases.

As mentioned in Section S1.2, considering columns of the input genotype matrix as binary vectors and sorting them in the non-decreasing order as a preprocessing step is expected to improve the prediction accuracy. As summarized in Table 3, this is well supported by the results presented in Figures 3 and 4. As shown in Table 3, the accuracy of our approach improves if the input genotype matrix is sorted by its columns. This

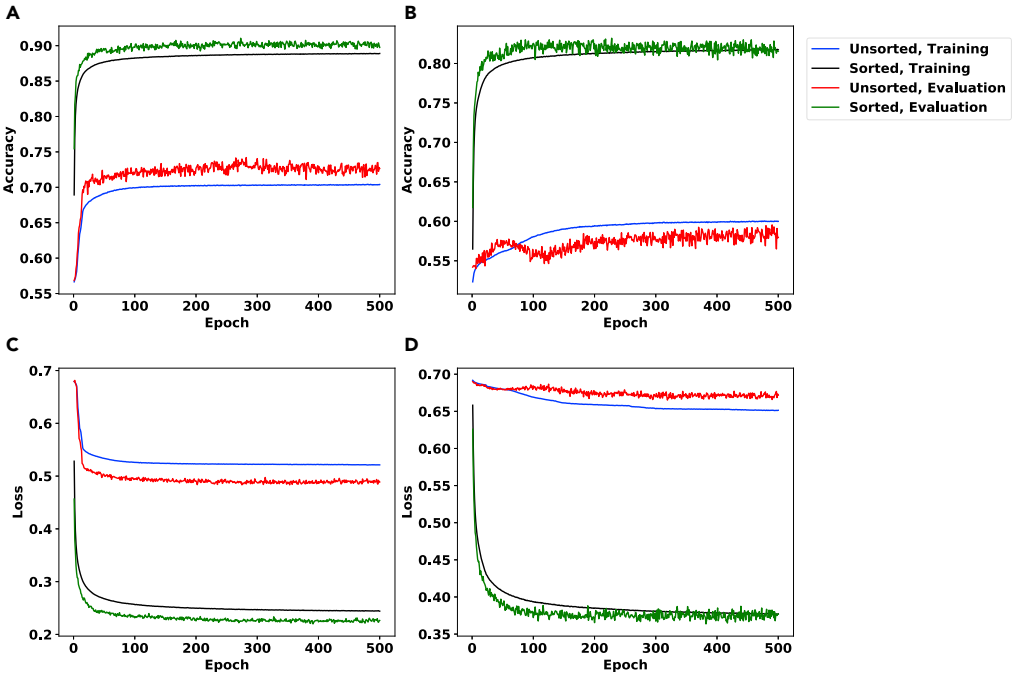


Figure 3. The Accuracy and Loss (Bishop, 2006) Plots for the *noise_inference* Problem Using the Neural Network Described in Section S1.2

The data sets in these experiments include matrices of size 10×10. The colors of the plots correspond to whether column-wise sorted or unsorted data sets were used – as well as whether the plot corresponds to the training phase or evaluation. We use α=0.002 and β=0.1 in panels (A,C) and $\alpha=4\times10^{-4}$ and β=0.02 in panels (B,D). Note that panels (A,B) depict accuracy, whereas panels (C,D) depict binary cross entropy loss (Bishop, 2006).

can be partially explained by the fact that sorting improves the locality of reference for mutations and makes it easier for the neural network to identify conflicts in a way reminiscent to the preprocessing step of the algorithm proposed by Gusfield (Gusfield, 1991).

In order to evaluate the relative advantage of neural networks against other machine learning techniques for the *noise_inference* problem, we explored the use of support vector machines (SVMs) (Bishop, 2006) for the same task. Figure 5 shows the training and evaluation accuracy for (column sorted) 10×10 matrices using SVMs.

The x axis in this figure represents the training sample size. Since the SVM is not memory efficient, the maximum number of input instances that can be used in training is limited by our computational resources. That is why the training procedure is stopped before the accuracy plot has converged. We use radial basis function (RBF) (Bishop, 2006) as the kernel in all of our experiments with SVMs. RBF is the most commonly used kernel in practice. In our experiments, it consistently showed better performance than other kernels (e.g., polynomial).

Observe that the accuracy of the neural network model on the evaluation data is notably higher than that for SVMs. E.g. the final evaluation accuracy values in Figure 3 vs. Figure 5 is 90% vs 73% in the first setting and 81% vs 64% in the second.

Noise Elimination

We train the network shown in Figure S1 using the reinforcement learning framework described in Section S1.3 on matrices of size 10×10, with $\alpha=4\times10^{-4}$ and β=0.02. We constructed an evaluation data set of 100 matrices with the α and β values identical to that of the training data set; although this noise profile is not currently observed in single-cell sequencing data sets, they have been achieved by emerging single clone sequencing data sets (Pérez-Guijarro et al., 2020).

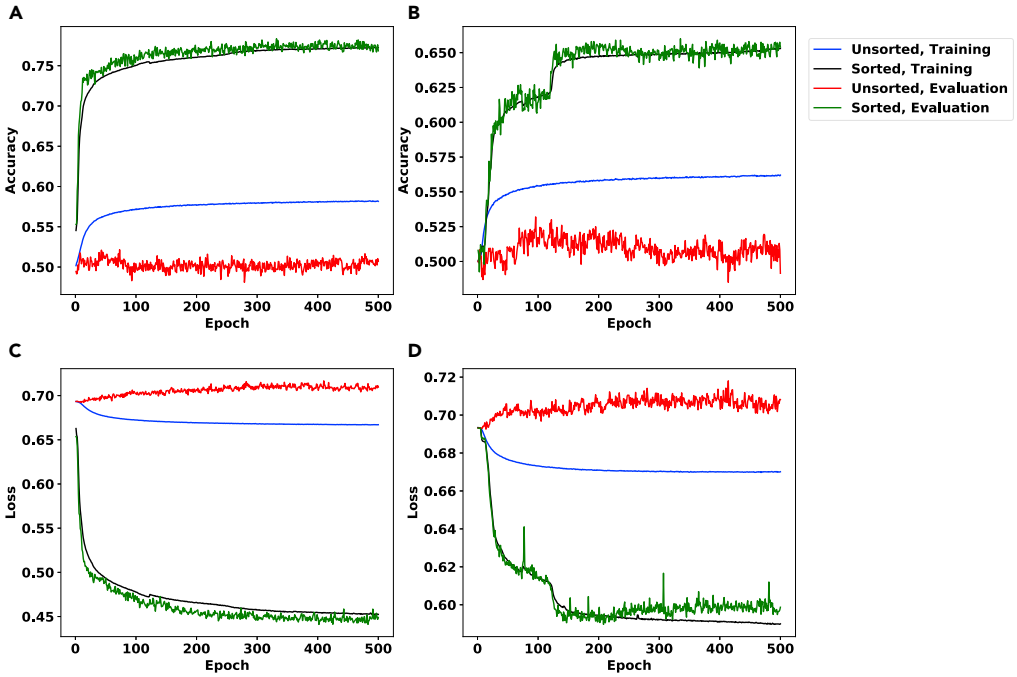


Figure 4. The Accuracy and Loss (Bishop, 2006) Plots for the *noise_inference* Problem Using the Architecture Described in Section S1.2 on 25×25 Matrices

The plots for column-wise sorted and unsorted genotype matrices, as well as those for training and evaluation phases, are depicted in distinct colors.

The noise settings for this figure are α=3.2×10^{-4} and β=0.016 for panels (A,C) and α=6.4×10^{-5} and β=0.0032 for panels (B,D). Note that panels (A,B) depict accuracy, while panels (C,D) depict binary cross entropy loss (Bishop, 2006).

We made sure that the training and evaluation data sets had no overlaps. Our trained model computed a conflict-free output matrix for 92% of the evaluation matrices.

Let o_{FP} and o_{FN} denote the number of flips from 0 to 1 and from 1 to 0, respectively, that we introduced to a ground truth conflict-free matrix A to produce A'. Similarly, let r_{FP} and r_{FN} denote the number of flips from 1 to 0 and 0 to 1, respectively, made by the model on the noisy input matrix A' in order to produce a conflict-free output matrix B. Figure 6A shows the histogram of $r_{FP}-o_{FP}$ (left side) and $r_{FN}-o_{FN}$ (right side) for the 92 instances of the problem for which our model produced a conflict-free matrix. Importantly, in the majority of cases, $r_{FP} \leq o_{FP}$ and $r_{FN} \leq o_{FN}$, indicating that the number of flips used by our approach is at most that of the number of flips introduced as noise (in some of the cases, there are solutions that result in a conflict-free matrix with fewer flips than that added as noise). We expand this visualization further in Figure 6B using a heatmap.

In order to quantify how well our approach's distribution of flip types compare to those flips introduced as noise, we introduce $ratio_{rl}=r_{FN}/(r_{FP}+r_{FN})$ and $ratio_{original}=o_{FN}/(o_{FP}+o_{FN})$. The x and y axis in Figure 7 represents $ratio_{rl}$ and $ratio_{original}$, respectively. Observe that most of the evaluation cases are on the line $ratio_{rl}=ratio_{original}$.

Next, we compare trees implied by the conflict-free matrices reported by our reinforcement learning approach with the trees obtained by running PhISCS (Malikic et al., 2019b). As input, we use the same data set that was used for experiments related to Figures 6 and 7. In all comparisons, we use recently developed multi-labeled tree similarity measure (MLTSM) for comparing trees of tumor evolution (Karpov et al., 2019). As shown in Figure 8, our approach achieves a tree reconstruction accuracy similar to PhISCS. It is worth noting that, on average, the reinforcement learning approach and PhISCS take 36.804 and 0.142 seconds, respectively, to process each matrix.

Finally, we evaluated how well our trained models generalize to input matrices with varying dimensionality. In Figure 9, we used a model trained on 10×10 matrices for this purpose. Observe that this trained model

Input Matrix Size	A	B	Unsorted Acc.	Sorted Acc.	Figure
10×10	0.002	0.1	72	90	Figure 3 (A)
10×10	4×10^{-4}	0.02	60	81	Figure 3 (B)
25×25	3.2×10^{-4}	0.016	50	77	Figure 4 (A)
25×25	6.4×10^{-5}	0.0032	52	65	Figure 4 (B)

Table 3. The Impact of the Preprocessing the Input Genotype Matrix by Sorting Its Columns as Described in Section S1.2 on the Evaluation Accuracy for the *noise_inference* Problem. The accuracy values for unsorted (fourth column) and column-sorted genotype matrices (fifth column) are reported as percentages rounded to the nearest integer.

can find a conflict-free matrix for a notable fraction of matrices with larger dimensionality. Note that the model found these conflict-free matrices by flipping at most 3 coordinates in each noisy matrix. As expected, our success rate decreases when the dimensionality increases.

Application to Real Data

We also applied our method for the *no_branching* problem to three real data sets for which previous phylogenetic analyses reported highly concordant trees of tumor evolution. The first data set consists of 16 single cells from a patient with triple-negative breast cancer (TNBC). Originally, it was made available in (Wang et al., 2014b), and here, we focused on the analysis of 18 mutations previously used in (Malikic et al., 2019a). The evolutionary history of these mutations was first (indirectly) reported in the original study, and the same results were later obtained by the use of B-SCITE (Malikic et al., 2019a). As shown in (Malikic et al., 2019a), the reported tree has high support from both single-cell and bulk data.

Second, we analyzed a dataset of a patient labeled as "Patient 6" from Acute Lymphoblastic Leukemia (ALL) study (Gawad et al., 2014), which consists of 146 single cells and 10 mutations. Similar to the TNBC data set, the evolutionary history of these mutations was thoroughly studied in the original study and later in (Kuipers et al., 2017).

Third, we applied our method to the data from "Patient 1", diagnosed with colorectal cancer that metastasized to the liver, published in (Leung et al., 2017) (note that in (Leung et al., 2017), this patient is also labeled as CRC1 or CO5). In total, 16 mutations were detected in the original study, and the evidence for presence of at least one of these mutations was found in 40 single cells from the primary (colon) site and 32 single cells from the metastatic (liver) site. Detailed analysis of the evolutionary history of this patient was also performed, and a tree of tumor evolution inferred by the use of SCITE (Jahn et al., 2016).

In order to apply our approach for the *no_branching* problem on these data sets, we trained four separate models with the same architecture described in Section S1.1. The dimensions of the input were set to 16×18 for the first data set and 146×10 for the second data set. We trained two models to apply to the third data set. The first model was trained with input dimensions set to 40×16 to process the submatrix corresponding to the cells from the primary site. The second model was trained with input dimensions set to 72×16 to process all cells (from both primary and metastatic sites).

In the training phase, only simulated instances (described in Section S1.1) were used. Note that one of the drawbacks for the model based on a feedforward neural network without any recurrent layer, described in Section S1.1, is that input dimensions are fixed. One can use padding or copying techniques to adjust the dimensions to any number of dimensions that a pre-trained model requires, as long as the number of dimensions of the input is smaller than that required by the model. However, this may have an adverse impact on the accuracy; training with the same number of dimensions typically leads to better results.

According to our results, the tree corresponding to the first data set contains at least one branching event (i.e., has non-linear topology), with probability 0.97. This result is highly consistent with previous analyses of this data set, wherein each analysis, several branching events in the tree of tumor evolution of this patient were reported (Malikic et al., 2019a, 2020; Ramazzotti et al., 2019; Singer et al., 2018; Wang et al., 2014b).

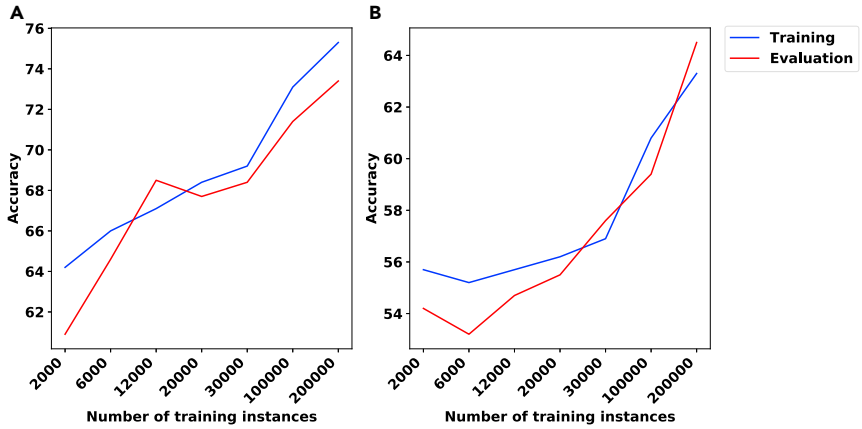


Figure 5. The Training and Evaluation Accuracy in the *noise_inference* Problem for SVM

All matrices for this experiment are of size 10×10 and are column sorted. The x axis corresponds to the training sample size.

(A) Training and evaluation accuracy for α=0.002 and β=0.1 (for the noisy matrices).

(B) Training and evaluation accuracy for $\alpha=4\times10^{-4}$ and β=0.02 (for the noisy matrices).

For the ALL patient, the probability of at least one branching event is only 0.21, suggesting a linear topology. This topology was also reported in the original study and later inferred by the use of SCITE (Jahn et al., 2016) in (Kuipers et al., 2017). Small but non-zero branching probability of 0.21 for this patient can be explained by some evidence for a recurrent mutation in gene SUSD2 presented in (Kuipers et al., 2017), which may potentially be due to a local branching event (involving only two mutations at the leaf level).

For the third data set, the tree reported in the original study (Leung et al., 2017) implies linear evolution at the primary tumor site (i.e., when we restrict our analysis to the cells originating from the primary site). However, when primary and metastatic cells are combined, the metastasis-specific mutations form a separate branch result in a tree with branching topology. In order to test whether our method can successfully distinguish the two cases, we run it on two different inputs: the first input consisting only of single cells sampled from the primary site and the second input consisting of cells from both primary and metastatic sites. Our models' probabilities of having linear topology reported on the first and the second inputs equal to 0.83 and 0.22, respectively. These values are highly consistent with the results presented in (Leung et al., 2017).

DISCUSSION

This paper introduces DL solutions to the three problems related to tumor phylogeny inference from SCS data, i.e., given a binary genotype matrix obtained from SCS data, with rows representing single cells and columns representing putative mutations, and false positive and false negative noise rates of the data: (1) Decide whether the most likely tumor phylogeny has a linear topology or contains one or more branching events, (2) Decide whether the matrix satisfies three gametes rule (and thus has a matching perfect phylogeny with genotypes of leaves of the phylogeny matching rows of the matrix), (3) Find the most likely tumor phylogeny based on the observed genotype matrix.

Note that there can be multiple equally likely solutions to the tumor phylogeny reconstruction problem and currently no available combinatorial technique can guarantee the inference of all of these solutions (in fact, all available techniques aim to infer a single most likely tumor phylogeny). Preliminary simulations on small data sets seem to suggest that the most likely tumor phylogeny inference problem possibly has a single optimal solution, provided the noise on the input SCS data is independently applied to the entries of the input genotype matrix. However, a large-scale study needs to be conducted to obtain a better understanding of the solution space.

Limitations of the Study

Our proposed approaches for the *no_branching* problem and *noise_inference* problem have the drawback that the input dimensions are fixed. We present a few remedies for this issue in "Application to Real Data".

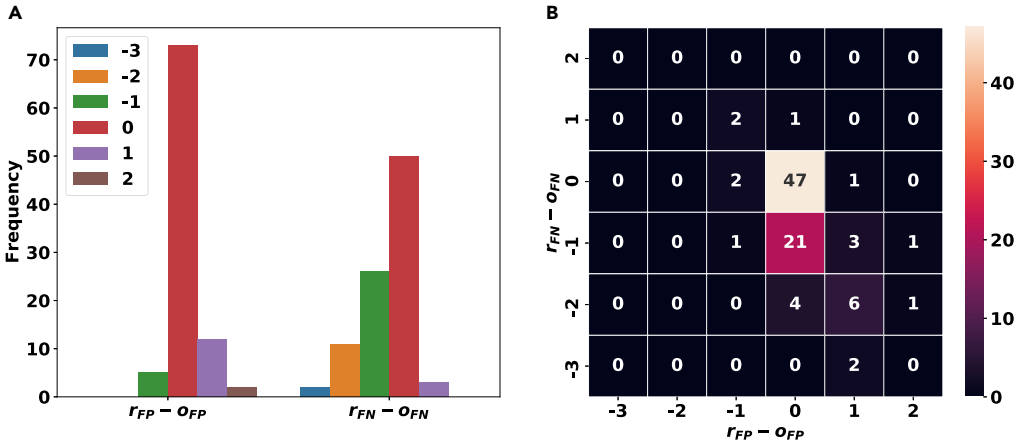


Figure 6. The Difference Between the Number of Flips Made by Our Approach and the Flips Introduced as Noise by the Simulation, i.e., to the Ground Truth to Obtain the Input Genotype Matrices, Given in Histograms in Panel (A) and a HeatMap in Panel (B), for the *noise_elimination* Problem

The left histogram in panel (A) and the x axis in panel (B) correspond to false positives. Similarly, the right histogram in panel (A) and the y axis in panel (B) correspond to false negatives. Lower values for $r_{FN}-o_{FN}$ and $r_{FP}-o_{FP}$ imply a good performance for our approach. The matrices used in this experiment are of size 10×10 and the noise rates are α=4×10^{-4} and β=0.02.

Furthermore, we did not perform any empirical running time comparison between our approach for the *noise_inference* problem and the linear-time algorithm introduced in (Gusfield, 1991), which is already asymptotically optimal. In addition, the real data sets explored in "Application to Real Data" for the *no_branching* problem were too large or their corresponding noise rates were too high for our noise

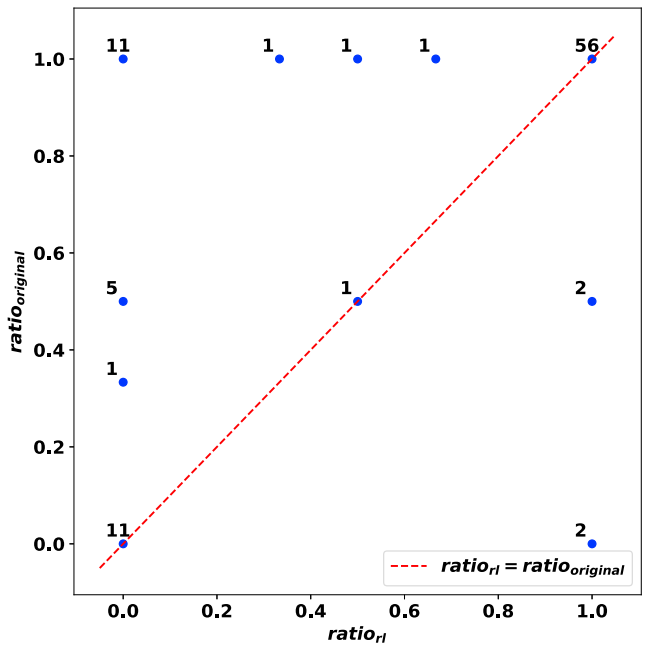


Figure 7. A Quantified Visualization of How Flips Made by Our Approach Correspond to those Added as Noise for the *noise_elimination* Problem

The plot compares the $ratio_{rl}$ with $ratio_{original}$ for matrices of size 10×10. Each blue dot is annotated by the number of respective evaluation cases. The red dotted line represents $ratio_{rl}$=$ratio_{original}$. The closeness of the blue dots for the majority of the evaluation cases to the red dotted line implies the recovery of the ratio of introduced flip types by our approach.

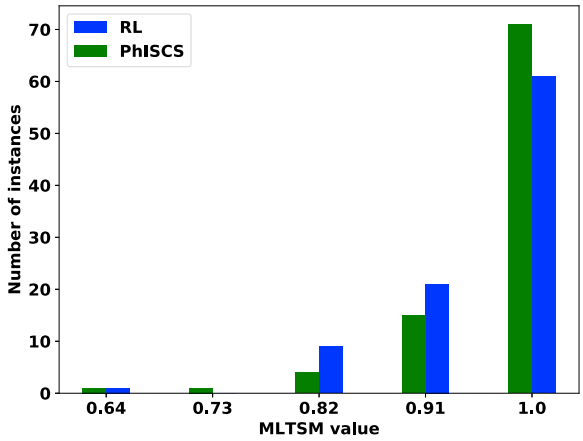


Figure 8. A Comparison of the Trees Reconstructed Using Our Reinforcement Learning Approach and the Trees Reported by PhISCS

MLTSM tree similarity measure is used in all comparisons. The y axis shows the number of instances for which MLTSM value shown on the x axis is obtained. The comparison is performed on the set of 92 matrices of size 10×10 with the false positive and false negative rates equal to 4×10^{-4} and 0.02, respectively.

elimination method, and thus, we were not able to include any results. Moreover, our approach does not improve the running time for obtaining a conflict-free matrix compared to the previous methods (e.g., PhISCS (Malikic et al., 2019b)).

Resource Availability

Lead Contact

Feel free to contact S. Cenk Sahinalp at cenk.sahinalp@nih.gov for further correspondence.

Materials Availability

Please see Supplemental Information online.

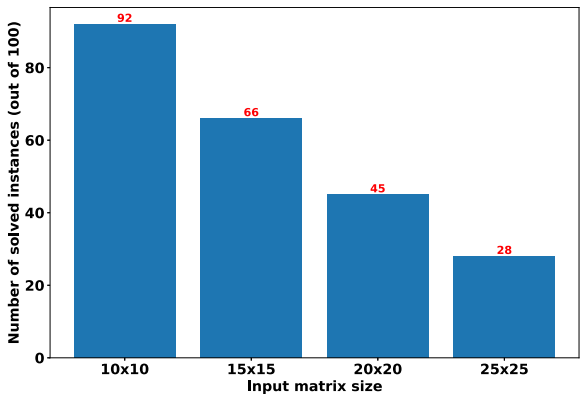


Figure 9. The Evaluation of Generalizability Across Different Values for Matrix Dimensions in the *noise_elimination* Problem

The model was trained on 10×10 matrices and was evaluated on larger matrices. Let solved instances here refer to those instances that the model finds a conflict-free matrix by flipping at some coordinates. We observed that the model flipped at most 3 coordinates for all solved instances in this experiment. The number of solved instances for each matrix dimension relative to that corresponding to 10×10 matrices can be thought of as a measure of generalizability. Noise rates for 10×10 matrices are $\alpha=4\times10^{-4}$ and $\beta=0.02$. Noise rates for the rest of the matrices are tuned in a way that we ensure the expected number of flipped entries stays fixed as the dimensions grow.

Data and Code Availability

The code for the construction of simulated data sets and running the training and evaluation for all of the networks described in this work is available at https://github.com/algo-cancer/PhyloM.

METHODS

All methods can be found in the accompanying Transparent Methods supplemental file.

SUPPLEMENTAL INFORMATION

Supplemental Information can be found online at https://doi.org/10.1016/j.isci.2020.101655.

ACKNOWLEDGMENTS

This work is supported in part by the Intramural Research Program of the National Institutes of Health, National Cancer Institute. This work was also supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute (Stewart et al., 2017). E.S.A. and S.C.S. were supported in part by the NSF grant AF-1619081, R.K. was supported in part by the NSF grant IIS-1906694, and M.H.E. was supported in part by Indiana U. Grand Challenges Precision Health Initiative.

AUTHOR CONTRIBUTIONS

S.C.S. identified the problems discussed in this work and foresaw the potentials for the use of machine learning methods for solving them. M.H.E. and E.S.A. with the help and supervision of S.C.S, S.M., and R.K., developed and implemented the methods and ran the experiments. S.M. also provided guidance on the real-world data set choice and design of the related experiments. R.K. advised the group on the development of machine learning methods and experiment design. All authors contributed to the writing of the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: May 20, 2020
Revised: August 10, 2020
Accepted: October 2, 2020
Published: November 20, 2020

REFERENCES

Bello, I., Pham, H., Le, Q.V., Norouzi, M. and Bengio, S. (2017), 'Neural combinatorial optimization with reinforcement learning', Workshop paper in International Conference on Learning Representations, ICLR .

Bishop, C.M. (2006). Pattern Recognition and Machine Learning (springer).

Bonizzoni, P., Ciccolella, S., Della Vedova, G., and Soto, M. (2018). Does relaxing the infinite sites assumption give better tumor phylogenies? an ilp-based comparative approach. IEEE/ACM Trans. Comput. Biol. Bioinformatics *16*, 1410–1423.

Chen, D., Eulenstein, O., Fernández-Baca, D., and Sanderson, M.J. (2006). Minimum-flip supertrees: complexity and algorithms. IEEE/ACM Trans. Comput. Biol. Bioinform. *3*, 165–173.

Ciccolella, S., Gomez, M.S., Patterson, M., Vedova, G.D., Hajirasouliha, I. and Bonizzoni, P. (2018), Gpps: an ilp-based approach for inferring cancer progression with mutation losses from single cell data, in '2018 IEEE 8th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS)', pp. 1–1.

Ciregan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In IEEE Conference on Computer Vision and Pattern Recognition (IEEE), pp. 3642–3649.

Deshwar, A.G., Vembu, S., Yung, C.K., Jang, G.H., Stein, L., and Morris, Q. (2015). Phylowgs: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. Genome Biol. *16*, 35.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: pre-training of deep bidirectional transformers for language understanding (NAACL-HLT).

Donmez, N., Malikic, S., Wyatt, A.W., Gleave, M.E., Collins, C., and Sahinalp, S.C. (2017). Clonality inference from single tumor samples using low-coverage sequence data. J. Comput. Biol. *24*, 515–523.

Eaton, J., Wang, J., and Schwartz, R. (2018). Deconvolution and phylogeny inference of structural variations in tumor genomic samples. Bioinformatics *34*, i357–i365.

Edrisi, M., Zafa, H., and Nakhleh, L. (2019). A Combinatorial Approach for Single-cell Variant Detection via Phylogenetic Inference. In 19th International Workshop on Algorithms in Bioinformatics (WABI 2019), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

El-Kebir, M. (2018). Sphyr: tumor phylogeny estimation from single-cell sequencing data under loss and error. Bioinformatics *34*, i671–i679.

El-Kebir, M., Oesper, L., Acheson-Field, H., and Raphael, B.J. (2015). Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. Bioinformatics *31*, i62–i70.

El-Kebir, M., Satas, G., Oesper, L., and Raphael, B.J. (2016). Inferring the mutational history of a tumor using multi-state perfect phylogeny mixtures. Cell Syst. *3*, 43–53.

Gawad, C., Koh, W., and Quake, S.R. (2014). Dissecting the clonal origins of childhood acute lymphoblastic leukemia by single-cell genomics. Proc. Natl. Acad. Sci. *111*, 17947–17952.

Gerstung, M., Jolly, C., Leshchiner, I., Dentro, S.C., Gonzalez, S., Rosebrock, D., Mitchell, T.J., Rubanova, Y., Anur, P., Yu, K., et al. (2020). The evolutionary history of 2,658 cancers. Nature *578*, 122–128.

Gusfield, D. (1991). Efficient algorithms for inferring evolutionary trees. Networks *21*, 19–28.

Gusfield, D. (1997). Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology (Cambridge University Press).

Hudson, R.R. (2002). 'Generating samples under a wright–Fisher neutral model of genetic variation'. Bioinformatics *18*, 337–338.

Husić, E., Li, X., Hujdurović, A., Mehine, M., Rizzi, R., Mäkinen, V., Milanič, M., and Tomescu, A.I. (2019). Mipup: minimum perfect unmixed phylogenies for multi-sampled tumors via branchings and ilp. Bioinformatics *35*, 769–777.

Jahn, K., Kuipers, J., and Beerenwinkel, N. (2016). Tree inference for single-cell data. Genome Biol. *17*, 86.

Karpov, N., Malikic, S., Rahman, M.K., and Sahinalp, S.C. (2019). 'A multi-labeled tree dissimilarity measure for comparing "clonal trees" of tumor progression'. Algorithms Mol. Biol. *14*, 17.

Kuipers, J., Jahn, K., Raphael, B.J., and Beerenwinkel, N. (2017). Single-cell sequencing data reveal widespread recurrence and loss of mutational hits in the life histories of tumors. Genome Res. *27*, 1885–1894.

Leung, M.L., Davis, A., Gao, R., Casasent, A., Wang, Y., Sei, E., Vilar, E., Maru, D., Kopetz, S., and Navin, N.E. (2017). Single-cell dna sequencing reveals a late-dissemination model in metastatic colorectal cancer. Genome Res. *27*, 1287–1299.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: a robustly optimized bert pretraining approach. arXiv *1907*, 11692.

Malikic, S., Jahn, K., Kuipers, J., Sahinalp, S.C., and Beerenwinkel, N. (2019a). Integrative inference of subclonal tumour evolution from single-cell and bulk sequencing data. Nat. Commun. *10*, 1–12.

Malikic, S., McPherson, A.W., Donmez, N., and Sahinalp, C.S. (2015). Clonality inference in multiple tumor samples using phylogeny. Bioinformatics *31*, 1349–1356.

Malikic, S., Mehrabadi, F.R., Azer, E.S., Ebrahimabadi, M.H., and Sahinalp, S.C. (2020). Studying the History of Tumor Evolution from Single-Cell Sequencing Data by Exploring the Space of Binary Matrices (bioRxiv).

Malikic, S., Mehrabadi, F.R., Ciccolella, S., Rahman, M.K., Ricketts, C., Haghshenas, E., Seidman, D., Hach, F., Hajirasouliha, I., and Sahinalp, S.C. (2019b). PhISCS: a combinatorial approach for subperfect tumor phylogeny reconstruction via integrative use of single-cell and bulk sequencing data. Genome Res. *29*, 1860–1877.

Myers, M.A., Satas, G., and Raphael, B.J. (2019). Calder: inferring phylogenetic trees from longitudinal tumor samples. Cell Syst. *8*, 514–522.

Pérez-Guijarro, E., Yang, H.H., Araya, R.E., El Meskini, R., Michael, H.T., Vodnala, S.K., Marie, K.L., Smith, C., Chin, S., Lam, K.C., et al. (2020). Multimodel preclinical platform predicts clinical response of melanoma to immunotherapy. Nat. Med. *26*, 781–791.

Popic, V., Salari, R., Hajirasouliha, I., Kashef-Haghighi, D., West, R.B., and Batzoglou, S. (2015). Fast and scalable inference of multi-sample cancer lineages. Genome Biol. *16*, 91.

Ramazzotti, D., Graudenzi, A., De Sano, L., Antoniotti, M., and Caravagna, G. (2019). Learning mutational graphs of individual tumour evolution from single-cell and multi-region sequencing data. BMC bioinformatics *20*, 1–13.

Ricketts, C., Seidman, D., Popic, V., Hormozdiari, F., Batzoglou, S., and Hajirasouliha, I. (2019). Meltos: multi-sample tumor phylogeny reconstruction for structural variants. Bioinformatics *36*, 1082–1090.

Ross, E.M., and Markowetz, F. (2016). Onconem: inferring tumor evolution from single-cell sequencing data. Genome Biol. *17*, 69.

Sadeqi Azer, E., Rashidi Mehrabadi, F., Malikić, S., Li, X.C., Bartok, O., Litchfield, K., Levy, R., Samuels, Y., Schäffer, A.A., Gertz, E.M., et al. (2020). Phiscs-bnb: a fast branch and bound algorithm for the perfect tumor phylogeny reconstruction problem. Bioinformatics *36* (*Supplement_1*), i169–i176.

Satas, G., and Raphael, B.J. (2017). Tumor phylogeny inference using tree-constrained importance sampling. Bioinformatics *33*, i152–i160.

Selsam, D., Lamm, M., Bünz, B., Liang, P., de Moura, L. and Dill, D.L. (2019), 'Learning a sat solver from single-bit supervision', poster paper in International Conference on Learning Representations, ICLR .

Senior, A.W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A.W., Bridgland, A., et al. (2020). Improved protein structure prediction using potentials from deep learning. Nature *577*, 706–710.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. Nature *550*, 354–359.

Singer, J., Kuipers, J., Jahn, K., and Beerenwinkel, N. (2018). Single-cell mutation identification via phylogenetic inference. Nat. Commun. *9*, 1–8.

Stewart, C.A., Welch, V., Plale, B., Fox, G., Pierce, M. and Sterling, T. (2017), Indiana University Pervasive Technology Institute.

Strino, F., Parisi, F., Micsinai, M., and Kluger, Y. (2013). Trap: a tree approach for fingerprinting subclonal tumor composition. Nucleic Acids Res. *41*, e165.

Wang, Y., Waters, J., Leung, M.L., Unruh, A., Roh, W., Shi, X., Chen, K., Scheet, P., Vattathil, S., Liang, H., et al. (2014b). Clonal evolution in breast cancer revealed by single nucleus genome sequencing. Nature *512*, 155–160.

Weber, L.L. and El-Kebir, M. (2020 (to appear)), Phyolin: Identifying a linear perfect phylogeny in single-cell dna sequencing data of tumors, in '20th International Workshop on Algorithms in Bioinformatics'.

Williams, R.J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learn. *8*, 229–256.

Wu, Y. (2020). Accurate and efficient cell lineage tree inference from noisy single cell data: the maximum likelihood perfect phylogeny approach. Bioinformatics *36*, 742–750.

Zaccaria, S., El-Kebir, M., Klau, G.W., and Raphael, B.J. (2017). The copy-number tree mixture deconvolution problem and applications to multi-sample bulk sequencing tumor data. In International Conference on Research in Computational Molecular Biology (Springer), pp. 318–335.

Zafar, H., Navin, N., Chen, K., and Nakhleh, L. (2019). Siclonefit: Bayesian inference of population structure, genotype, and phylogeny of tumor clones from single-cell genome sequencing data. Genome Res. *29*, 1847–1859.

Zafar, H., Tzen, A., Navin, N., Chen, K., and Nakhleh, L. (2017). Sifit: inferring tumor trees from single-cell sequencing data under finite-sites models. Genome Biol. *18*, 178.

iScience, Volume *23*

Supplemental Information

Tumor Phylogeny Topology Inference via Deep Learning

Erfan Sadeqi Azer, Mohammad Haghir Ebrahimabadi, Salem Malikić, Roni Khardon, and S. Cenk Sahinalp

Supplemental Information
Tumor Phylogeny Topology Inference via Deep Learning

Erfan Sadeqi Azer, Mohammad Haghir Ebrahimabadi, Salem Malikić, Roni Khardon, S. Cenk Sahinalp

S1 Transparent Methods

S1.1 Solutions to the Branching Inference Problem

Training Dataset. For this problem, we first generate a set of conflict-free matrices such that $\mathsf{Sing}(A) = 0$ for each matrix A from this set (i.e., tree implied by A has a branching topology). We generate this set through the procedure explained in Section 2. In order to generate a second complementary set of conflict-free matrices, for each matrix A from the first set, we describe below how a new random matrix A^L can be constructed with the same number of ones, but $\mathsf{Sing}(A^L) = 1$, i.e., A^L satisfies the no-branching property. This construction not only guarantees the union of these two sets have a balanced number of instances for branching and non-branching (linear) topologies, but also eliminates any *artificial* distinguishing features (e.g., based on the number of ones in the matrix, which is expected to be higher for matrices implying topologies with a lower number of branching events).

Briefly, consider an arbitrary matrix $A_{n\times m}$ from the set of matrices satisfying no-branching property generated above. Assume that A contains exactly l entries equal to 1. In order to generate related matrix A^L we first produce a random non-decreasing sequence consisting of n non-negative integers $u_1, u_2, \ldots, u_n$ which add up to l and each is upper-bounded by m. For each $i \in \{1,2,\ldots,n\}$ and $j \in \{1,2,\ldots,(m-u_i)\}$ we set $A^L[i,j] = 0$. All other entries of A^L are set to 1. The matrix A^L generated using this procedure obviously contains exactly l ones. Furthermore, A^L is also a staircase matrix and therefore it has the no-branching property.

Before adding it to our training dataset, we modify the matrix A^L generated above by randomly permuting its rows and columns (this obviously does not affect the topology implied by A^L nor the number of ones that it contains).

We combined the two sets of matrices and used them for training without adding noise. However, we evaluated the trained model on both noisy and noise-free inputs.

Network Structure. A two-layer fully connected artificial neural network is used with either 10 or 100 hidden neurons. The input layer corresponds to an arbitrary-size batch of vectors with length $n \cdot m$. We used `tanh` as the activation function. To avoid overfitting, a drop-out (Srivastava et al. 2014) rate of 0.9 was used. Note that the architecture had a moderate level of sensitivity to the hyperparameter settings (i.e., the number of neurons and the drop-out rate). The output of this network is one value indicating the probability of the input matrix having no-branching property. During training, a binary cross entropy loss (Bishop 2006) is used to compare the predictions against binary labels given as the ground truth.

S1.2 Solutions to the Noise Inference Problem

In order to solve the noise_inference problem, we formulated it as a classification task through supervised learning.

Training Dataset. As described in Section 2, we first generated a set of conflict-free matrices A, i.e., $\mathsf{lcf}(A) = 0$, and then for each matrix A we obtained its noisy version A' with predetermined noise parameters α and β so that $\mathsf{lcf}(A') = 1$. The combined set of matrices A and A' forms a balanced binary dataset of noisy and noise-free matrices as required for classification purposes.

For some of our experiments, we preprocessed each input matrix as per (Gusfield 1991) by sorting its columns in non-decreasing order based on the binary number each represents (read from top to bottom). E.g., this would move column $\begin{bmatrix}0\\1\\1\end{bmatrix}$ to the left of column $\begin{bmatrix}1\\0\\0\end{bmatrix}$ as $(011)_2 = 3 < (100)_2 = 4$. Sorting columns was shown to help checking whether a matrix is conflict-free or not in non-ML settings (Gusfield 1991). We show that it also helps ML strategies (see Section 3).

Network Structure. For this problem, we use a two-layer fully connected network with 100 neurons in each layer. The input layer corresponds to an arbitrary-size batch of vectors with length $n \cdot m$. The Sigmoid function was used as the activation function for both layers. We used a drop-out rate of 0.2 for each layer for preventing our network from overfitting. The output of this network is one value indicating the probability of the input matrix being conflict-free. During training, a binary cross entropy loss (Bishop 2006) is used to compare the predictions against binary labels given as the ground truth.

S1.3 Solutions to the Noise Elimination Problem

This section presents our approach for solving the noise_elimination problem and thus contains our major contributions from a methodological standpoint.

S1.3.1 Input Format

Recall that input to this problem is given as a binary matrix A'. We transform A' to a new matrix A'' as shown below, before feeding it to the neural network. Each row of A'' corresponds to exactly one of the entries of A'. The first two columns of a given row in A'' respectively represent the row and the column of the corresponding entry in A'. The third column represents the noise level and depends on the actual value of the entry - which is represented in the last column. In the simplest case, the value of the third column is either α - if the entry has value one, or is β - if the entry has value zero. In the more general case, the user can specify a distinct false positive or false negative rate for each entry of the input matrix A'. As depicted below, this transformation is key to our ability for training the neural network with matrices of varying shapes/dimensions:

$$\underbrace{\begin{bmatrix} a'_{1,1} & a'_{1,2} & \dots & a'_{1,j} & \dots & a'_{1,m} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a'_{i,1} & a'_{i,2} & \dots & a'_{i,j} & \dots & a'_{i,m} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a'_{n,1} & a'_{n,2} & \dots & a'_{n,j} & \dots & a'_{n,m} \end{bmatrix}}_{A'_{n\times m}} \rightarrow \underbrace{\begin{bmatrix} 1 & 1 & f(a'_{1,1}) & a'_{1,1} \\ 1 & 2 & f(a'_{1,2}) & a'_{1,2} \\ \vdots & \vdots & \vdots & \vdots \\ i & j & f(a'_{i,j}) & a'_{i,j} \\ \vdots & \vdots & \vdots & \vdots \\ n & m & f(a'_{n,m}) & a'_{n,m} \end{bmatrix}}_{A''_{n\cdot m\times 4}}$$

where,

$$f(a'_{i,j}) = \begin{cases} \alpha & \text{if } a'_{i,j} = 1 \\ \beta & \text{if } a'_{i,j} = 0. \end{cases} \tag{4}$$

Note that the values of α and β are typically estimated as part of single-cell data analysis (e.g., by considering coverage across heterozygous sites) (Gawad et al. 2014, Leung et al. 2017, Wang et al. 2014a). While the value of β in most of the available datasets is within the range from 0.1 to 0.3, the value of α is usually lower than 0.01.

S1.3.2 Method

Our approach is similar to the algorithm in (Bello et al. 2017), which follows the model-free policy-based reinforcement learning framework, and was developed to solve TSP and knapsack problems. This framework requires a *cost function* to be defined on the space of potential outputs for the network. In the training phase, the optimization algorithm tries to minimize this cost function. Notice that this will be the only source of supervision we provide to our model for training purposes. Importantly, we do not need to know or calculate through alternative methods the ground truth A, which is more expensive to infer in comparison to calculating our cost function as described below.

The cost function we came up with is defined for a given input matrix A' and a potential output matrix X, with entries $x_{i,j}$, and has two terms. The first term, $\mathsf{NumberOfConflicts}(X)$ is equal to the number of distinct conflicts

present in X. As such, it will have value 0 if X is conflict-free:

$$\begin{aligned}\mathsf{NumberOfConflicts}(X) = \sum_{c_1\in[m],c_2\in[m],c_1<c_2} \Big(& \sum_{r_1\in[n]} [(x_{r_1,c_1},x_{r_1,c_2}) = (0,1)]\cdot \\ & \sum_{r_2\in[n]} [(x_{r_2,c_1},x_{r_2,c_2}) = (1,0)]\cdot \\ & \sum_{r_3\in[n]} [(x_{r_3,c_1},x_{r_3,c_2}) = (1,1)]\Big).\end{aligned} \tag{5}$$

In addition to the $\mathsf{NumberOfConflicts}(X)$, we use the negative log-likelihood of X, introduced in Equation (2), as the second term of the cost function $\mathsf{Cost}(X) : \{0,1\}^{n\times m} \to \mathbb{R}^+ \cup \{0\}$, which is defined below:

$$\mathsf{Cost}(X) = \mathsf{NumberOfConflicts}(X) - \gamma \cdot \ln(P(A'|A = X, \alpha, \beta)) \tag{6}$$

where $\gamma > 0$ is a hyper-parameter for establishing a trade-off between the two terms.

Recall that the objective of our optimization problem is to compute a matrix with the highest likelihood among all matrices with $\mathsf{NumberOfConflicts}(X) = 0$. Equation 6 achieves this goal when γ is set to a small value (e.g., it can be easily shown that any value that is smaller than $-\frac{1}{2n\ln\beta}$ is sufficiently small to be set as a value of γ).

The core architecture we use here, within the reinforcement learning framework, is a *pointer* network (Bello et al. 2017). Figure S1 shows the components of the pointer network and the flow of information when it is used for evaluation, as summarized below. The additional connections used for training, e.g., for back-propagation, are not shown in the figure.

The embedding layer embeds the input array of shape $(n \cdot m) \times 4$ into an array of shape $(n \cdot m) \times d$, where d is the embedding length used in the network. The encoder consists of q layers of convolution (Fukushima 1980). Each one of the $(n \cdot m)$ rows (with length d) of the embedded array goes through the q convolution layers to produce a length d vector placed in the buffer as a new row. In our experiments, the values of d and q were set to 8 and 2, respectively. The decoder is a long short-term memory (LSTM) (Hochreiter & Schmidhuber 1997) layer and the attention layer (Bahdanau et al. 2015) is a network of fixed size. Together they compute the output of the neural network. Specifically, the attention layer iteratively calculates a real valued *score* function on d dimensional vectors. In each iteration, this function is applied to each row of the buffer to obtain its score, which is then used to pick a single row from the buffer (corresponding to a row of the input to be flipped), to be fed to the LSTM. The output of the LSTM is another vector of length d, which is used to update its memory and is then fed to the attention layer, to be used to update the score function. The cost function we described earlier is employed both in updating the LSTM as well as the score function of the attention layer. Note that the score function is not trivially applied only to the rows of the buffer that have not been picked earlier. Those that have been picked are automatically assigned a fixed low score. A soft-max function is then used on these scores to get a probability distribution over the rows of the buffer. In the next iteration, the row to be fed to LSTM is sampled according to this distribution.

This neural architecture is used to decide and evaluate a set of proposed flips through the cost function. More specifically, the buffer row selected by the attention mechanism at each iteration is considered to be the proposed flip location. We combine all the proposed flips to A' to get the output matrix B. Then the value of $\mathsf{Cost}(B)$ is calculated and used as feedback for the network. Since the architecture is trained via policy gradients, the total cost function is sufficient as the feedback signal, and the learned policy, implicit in the network and attention mechanism, will attempt to minimize the cost of the output B.

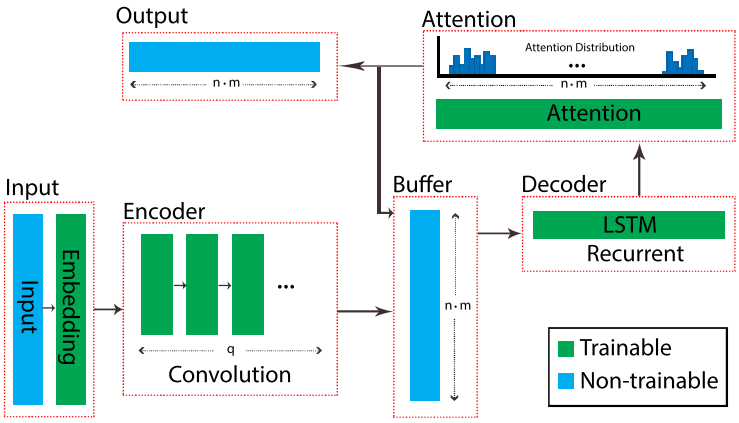


Figure S1: The architecture of the network used for the noise_elimination problem, Related to Section 2.3 and Table 1. The roles for the inner-components of the network including embedding layer, encoder, decoder, and attention mechanism interacting with the buffer, are explained in the text. In each training epoch, a batch of input matrices is fed to the network. Each input matrix $A'_{n\times m}$ in a batch, is preprocessed to an $(n \cdot m) \times 4$ matrix as described in Section S1.3.1. The output of the attention layer, right before the final output layer, is a function that forms a distribution over the $n \cdot m$ entries of the input matrix A' (after having been processed through the embedding layer and the encoder). The attention layer iteratively forms the final **output** of the network, which is a set of entries of A' to flip.

S2 Experimental setup

All the experiments in this work are performed using Carbonate, a computer cluster at Indiana University. We used deep learning (DL) nodes in Carbonate. These nodes feature 12 GPU-accelerated Lenovo ThinkSystem SD530, each equipped with two Intel Xeon Gold 6126 12-core CPUs, two NVIDIA GPU accelerators (eight with Tesla P100s; four with Tesla V100s), and 192 GB of RAM. A Red Hat Enterprise Linux Server 7.7 is running on these nodes, and we used Python 3.7.3 :: Anaconda, Inc., Tensorflow 1.13.1, and Keras 2.2.4.

S3 Details of simulating genotype matrices

In order to generate genotype matrices used as the input, we first run ms simulator (Hudson 2002), which generates conflict-free matrices under the infinite sites model. We run the following command to generate conflict-free matrices:

```
>>> ms nsam nreps -s
```

where,

- nreps denotes the number of independent samples to generate (the number of conflict-free matrices)
- nsam denotes the number of copies of the locus in each sample (the number of cells)
- $-$s denotes the number of segregating sites (the number of mutations)

Let A denote a conflict-free matrix generated by ms. In order to obtain instances of a noisy matrix A' used in our experiments, we *added noise* to A by *flipping* some of its entries. For each entry which equals 1 in A, we decide whether to flip it or not by a simple draw from the Bernoulli distribution with the probability of success equal to the false negative error rate of single-cell sequencing experiment. In case of success we set $A'[i,j] = 0$, otherwise $A'[i,j] = 1$. Entries $A'[i,j]$ for pairs (i,j) for which $A[i,j] = 0$ are obtained analogously by considering false positive error rate of single-cell sequencing experiment as the success probability in the Binomial distribution.

Recall that we also check each matrix A' labeled as a matrix containing a conflict to verify that it has at least one *conflict*. If it does not, the above procedures are repeated until a new matrix A' containing conflict is produced.

S4 Discussion of the sensitivity of the experiments to hyperparameters

There are a few parameters in our methods that can take values from their domain sets. We run an exploratory experiment and observe that there is no high sensitivity when these parameters are changed. Here, we provide a brief discussion of three main parameters.

γ in cost function. We performed some preliminary experiments (not mentioned in Section 3.4) on matrices with size 10×10 for γ in range $[0.004, 4]$ to explore the impact of this parameter on accuracy. We observed that with $\gamma = 0.004$ the accuracy is 86%. Similarly for γ with values 0.04, 1, and 4 the accuracy is 83%, 80%, and 84% respectively.

Drop-out rate. The neural networks in Sections S1.1 and S1.2 use drop-out technique to avoid overfitting. We tried several values from the range $[0.1, 0.9]$. We choose to use 0.9 within the networks used in Section S1.1, and 0.2 in Section S1.2. For example in the noise_inference problem using our sorted dataset of size 10×10 with $\alpha = 0.002$ and $\beta = 0.1$, when the activation function was Sigmoid and the model was trained only for 20 epochs, for drop-out rates of 0.2, 0.5, and 0.8 the evaluation accuracies 88%, 85%, and 78% could be achieved respectively.

Activation Function. We tried common activation functions, e.g., ReLU, tanh, and Sigmoid, within the neural networks in Sections S1.1 and S1.2. We choose to use tanh within the network in Section S1.1, and Sigmoid in Section S1.2. We observe that these two functions are generally performing better than ReLU. For example in the noise_inference problem using our 10×10 sorted dataset with $\alpha = 0.002$ and $\beta = 0.1$ we achieved 82% and 80% evaluation accuracy while ReLu and tanh were used as activation functions, respectively. For the dataset with the same matrices but unsorted, the evaluation accuracy was 68% when ReLu is used and 67% when tanh is used as the activation function.

S5 Training time

In this section, we report the time it takes for training each of three neural networks that correspond to the three problems that we attempt to solve in this work.

Branching Inference. The training time for $2K$ matrices, including $1K$ from each class of size 100×100 for 200 epochs, was 995.21 seconds on our machines. With the same dataset size and the number of epochs, it took 1547 seconds (around 25 minutes) to train matrices with 500×500 dimensions.

Noise Inference. For this problem training on $2M$ matrices, $1M$ each class, of size 10×10 for 500 epochs took 65940 seconds. For the same number of training and evaluation matrices of size 25×25 and the same number of epochs, the running time was 67979 seconds. It is worth mentioning that noise rates do not affect the training time.

Noise Elimination. In this problem, the network was trained on batches consisting of 128 unique matrices. On each batch, the network was trained only once. The training time for all 14000 batches (i.e., 1792000 matrices in total) was 52380 seconds.