Towards better prediction of Mycobacterium tuberculosis lineages from MIRU-VNTR data - Supplementary Materials

Nithum Thain[a], Christopher Le[a], Aldo Crossa[b], Shama Desai Ahuja[b], Jeanne Sullivan Meissner[b], Barun Mathema[c], Barry Kreiswirth[d], Natalia Kurepina[d], Ted Cohen[e], Leonid Chindelevitch[a]

[a]*School of Computing Science, Simon Fraser University, Burnaby, BC, Canada*
[b]*New York City Department of Health and Mental Hygiene, Queens, NY, USA*
[c]*Department of Epidemiology, Mailman School of Public Health, Columbia University, New York, NY, USA*
[d]*Public Health Research Institute TB Center, Rutgers University, Newark, NJ, USA*
[e]*Epidemiology of Microbial Diseases, Yale School of Public Health, New Haven, CT, USA*

1. The LP and ILP formulations and NP-hardness proof

1.1. LP and ILP formulations

The linear programming (LP) formulation of the problem of determining the smallest set of rules that can accurately predict those strains belonging to a given lineage (denoted by P, and called "positive examples") from those that do not belong to it (denoted by N, and called "negative examples"), is adopted from the one proposed by Malioutov and Varshney [1].

Let n be the total number of strains and l, the total number of loci. We start by exchanging the roles of positive and negative examples and define $\vec{y} \in \{0,1\}^m$ as a vector with value 1 in the components corresponding to negative examples, and 0 in those corresponding to positive examples. We also define the $m = 2 \cdot l \cdot K$ variables x_{jb} and y_{jb}, where K is the largest copy number a locus can have. The variable x_{jb} is true for a strain when it has value more than b in locus j, while the variable y_{jb} is true for it if it has value at least b in locus j. Note that exactly half of the variables are true for each strain. Lastly, we define the matrix $A \in \{0,1\}^{m \times n}$ whose rows correspond to the strains and whose columns correspond to the variables x and y. We

define $A_{ij} = 0$ if variable j is true for strain i, and 1 otherwise. Note that

$$\vec{y} = A \bigvee \vec{w} \iff \vec{y}^C = A^C \bigwedge \vec{w},$$

where $\bigwedge$ and $\bigvee$ denote the Boolean (logical) conjunction and disjunction, respectively, and $\vec{w}$ is any Boolean (0-1) vector. For this reason, since we are looking for conjunctive rules, we defined both $\vec{y}$ and A as the complemented versions of what we would normally expect them to be.

The ILP formulation given in [1] is

$$\begin{aligned} \text{Minimize } & \sum_{j=1}^{n} w_j \text{ subject to} \\ & w_j \in \{0,1\} \; \forall \; j \\ & A_P \vec{w} \geq \vec{1} \\ & A_N \vec{w} = \vec{0}, \end{aligned}$$

where P and N denote the subsets of positive ($\vec{y} = 0$) and negative ($\vec{y} = 1$) examples, respectively, and $\vec{w}$ is a sparse vector containing a 1 for those variables that will form the complex rule.

Since this problem is generally not solvable if the positive examples are not perfectly separable by a complex rule from the negative ones, Malioutov and Varshney use slack variables to penalize errors, as follows:

$$\begin{aligned} \text{Minimize } & \sum_{j=1}^{n} w_j + \lambda \sum_{i=1}^{m} \xi_i \text{ subject to} \\ & w_j \in \{0,1\} \; \forall \; j \\ & \xi_i \geq 0 \; \forall \; i \\ & \xi_i \leq 1 \; \forall \; i \in P \\ & A_P \vec{w} + \xi_P \geq \vec{1} \\ & A_N \vec{w} = \xi_N, \end{aligned}$$

where λ is a penalty parameter and the ξ_i are so-called slack variables.

This is the ILP formulation that we optimize, with $\lambda = 100$ chosen to be the penalty parameter throughout our experiments. On the other hand, the original paper [1] further relaxes this to an LP by replacing the Boolean (integrality) constraint on $\vec{w}$ with the inequality $0 \leq w_j \leq 1 \; \forall \; j$.

1.2. The rule extraction process and classification strategy

We adopt an iterative strategy for extracting rules for the set of lineages we have. At every iteration, the rule that gets added is the one that has the lowest weighted sum of the cost of the rule and the penalty term; all strains with the corresponding lineage are then removed from the dataset. The process is repeated until only two lineages remain; the rules produced at this last stage are taken to be those that differentiate the remaining two lineages from one another.

To classify a new strain, the rules are applied in the same order that they were produced, and the first one that the strain satisfies is the one that determines the lineage that gets assigned to it. We experimented with two variants of this strategy - a *hard* classification, and a *soft* classification. In the former, the first rule that is satisfied determines the lineage of the strain, and if the strain does not satisfy any rule, then no lineage is assigned. In the latter, each lineage gets a score that reflects the fraction of the simple rules in the complex rule that it satisfies, and the lineage with the highest score gets selected, with ties resolved in favor of earlier rules. Our experiments showed that the soft strategy tends to work better, and it is the one we adopted here.

2. Description of the machine learning methods used

2.1. Random Forest

A **decision tree** is a machine learning classifier which is often favored for its interpretability and robustness [2]. Building a decision tree consists of recursively splitting a dataset by identifying threshold values of variables which are successful at separating the data by class. Two often used criteria for measuring the effectiveness of a given split are the *Gini Index* and the *entropy* [3]. Once a tree is built, predicting the class of a data point simply consists of comparing the point to each of the variable thresholds along the tree, starting from the root, and arriving at a leaf node labeled with a class.

While decision trees have some advantages, they tend to be vulnerable to overfitting and produce classifiers with high variance. In order to reduce this variance, a **random forest** is built by collecting a large number of decision trees. Oftentimes, each tree is trained on both a different subset of the data and a subset of the available variables to increase variability and reduce overfitting. When these trees are combined, the ultimate prediction obtained by taking the mode among the classes predicted by different decision trees tends to have a higher accuracy and significantly lower variance than that of any individual tree.

2.2. Gradient Boosted Machine

A **gradient boosted machine** is a machine learning technique which combines multiple weak learners into a single strong learner [4]. Typical examples of weak learners include logistic regression models or decision trees. Unlike random forests, gradient boosted machines don't simply train these weak learners independently and aggregate them. Rather, each learner is trained iteratively to improve the classification strength of the aggregate. In particular, initial learners help determine which examples are hard and later learners focus on getting these hard examples right. Because of this gradient boosted machines can have a higher accuracy than random forests but are more vulnerable to overfitting.

2.3. k-Nearest Neighbors

The k **nearest neighbors** algorithm is among the simplest of machine learning algorithms [5]. Given a training set, this algorithm classifies new data points by majority vote of the classes of the k closest data points in the training set. The definition of closest depends on the metric that is chosen. Popular metrics include the Euclidean distance (L_2 norm) or rectilinear distance (L_1 norm) [6]. One weakness of the algorithm is that more frequent classes tend to dominate the neighborhoods of less frequent classes, which can lead to misclassification when the underlying distribution is skewed. This can be corrected somewhat by weighing the votes of the neighbors by how far away each one is from the data point of interest.

The value of k is a hyper-parameter of this model and can affect its overall performance. Using 3-fold cross-validation, we determine that using a value

of $k = 5$ maximizes our model's accuracy. Note that the standard method for predicting lineages from MIRU-VNTR data can be seen as a k nearest neighbors algorithm with $k = 1$.

2.4. Multinomial Logistic Regression

Multinomial logistic regression is an extension of the logistic regression algorithm to the multiclass setting, i.e. where the data can belong to more than two possible classes [7]. This technique works by modeling the log probability that a data point belongs to a certain class as a linear combination of the variables minus a normalization factor. The coefficients in this linear model are learned from the training data, in a manner analogous to standard logistic regression. The model additionally assumes that each variable has a single value for each class and that the collinearity between variables is low.

3. Sensitivity analysis

Table 1 shows the results of repeating our experiments with 100 random partitions of the data into an 80% training set and a 20% testing set. The values in brackets are the standard deviations. As we can see by comparing it to table 5, our results on the random partition we used are consistent with the observed distribution, and in fact mostly fall within 1 standard deviation, and always within 3.6 standard deviations from the mean. Furthermore, almost all of the z-scores with large magnitude are negative, suggesting that the results we report in table 5 are generally conservative.

We also compared the sets of rules produced by RuleTB with different splits of the data, in order to make sure that they are generally pretty consistent. In order to do so, we compared a) the order in which the rules were extracted and b) the overlap between the sets of rules used to predict each lineage. We found that 87 out of 100 splits had the exact same rule order as the one we found, 10 had the order of the last two rules reversed, and 3 had the order of the penultimate and the antepenultimate rules reversed, suggesting that the order we found is quite robust. Furthermore, by computing the Jaccard index [8] between the sets of rules we found for each lineage, we found a mean index of 0.44 (standard deviation 0.09) between the rules we extracted with the original split and those for the other splits, suggesting

Algorithm	Overall	EAI	East-Asian	Euro-American	Indo-Oceanic	M. africanum	M. bovis
MIRU-VNTRplus ($\star$)	49.1 (2.1)	45.1 (7.6)	74.4 (4.7)	38.1 (2.8)	55.8 (6.2)	70.6 (13.4)	92.8 (10.8)
Same, no threshold ($\star$)	98.5 (0.6)	96.1 (2.8)	98.7 (1.0)	99.0 (0.6)	98.6 (1.5)	91.5 (8.2)	100 (0)
MIRU-VNTRplus (†)	91.8 (1.1)	95.8 (2.8)	97.5 (1.7)	93.0 (1.5)	78.6 (4.9)	87.7 (10.4)	79.8 (17.6)
Same, no threshold (†)	97.7 (0.6)	98.3 (1.7)	99.4 (0.8)	98.6 (0.7)	91.5 (3.5)	94.9 (5.7)	100 (0)
TB-Insight	98.3 (0.6)	92.8 (3.9)	98.8 (1.2)	99.1 (0.6)	99.0 (1.3)	91.5 (8.2)	100 (0)
TBminer - MIRU-VNTRplus	98.2 (0.6)	91.6 (4.3)	98.8 (1.1)	99.1 (0.6)	99.0 (1.3)	91.5 (8.2)	100 (0)
TBminer - Expert	98.2 (0.6)	95.5 (3.2)	99.0 (1.0)	99.0 (0.6)	97.4 (2.0)	86.3 (10.9)	100 (0)
TBminer - MIRU-VNTRplus (‡)	97.7 (0.7)	89.6 (4.6)	98.2 (1.3)	99.1 (0.6)	99.0 (1.3)	85.7 (10.6)	97.0 (7.2)
TBminer - Expert (‡)	97.5 (0.6)	89.6 (4.6)	98.3 (1.3)	98.8 (0.7)	98.6 (1.5)	82.1 (11.4)	100 (0)
RuleTB	94.9 (1.1)	93.4 (5.5)	93.2 (4.3)	95.8 (1.1)	96.0 (2.5)	88.9 (9.8)	86.2 (11.6)
StackTB	98.8 (0.5)	97.4 (2.4)	99.7 (0.6)	99.0 (0.6)	98.7 (1.5)	91.5 (8.2)	99.2 (3.9)

Table 1: Mean (standard deviation) accuracy of algorithms trained on broad lineages over 100 random partitions. Legend: ($\star$) = the MIRU-VTNRplus database is used for training; (†) = the training subset of the entire data is used for training; ‡= only the 15 most discriminative VNTR loci are used for the prediction.

that these rules themselves are more often than not helpful for other splits (at even odds the Jaccard index, computed as $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$, would be $\frac{1}{3}$).

A qualitatively similar situation occurs for the other two datasets, where we also create 100 random partitions of the data, and report the mean and standard deviation of the resulting distribution of accuracies in Tables 2 and 3, respectively. The increasing number of lineages resulted in an increase in the number of possible orders in which the rules were extracted, with 5 different orders on the refined lineages and 7 on the SNP clusters. However, the Jaccard index remained high, with a mean of 0.47 and a standard deviation of 0.08 in both cases, suggesting robustness of the found rules.

Algorithm	Overall	EAI	East-Asian	Euro-American	Indo-Oceanic	M. africanum 1	M. africanum 2	M. bovis	M. caprae
MIRU-VNTRplus ($\star$)	49.5 (2.1)	44.9 (7.1)	75.0 (4.3)	38.2 (3.0)	56.7 (5.2)	53.4 (16.5)	100 (0)	93.4 (10.7)	100 (0)
Same, no threshold ($\star$)	97.8 (0.6)	95.5 (2.8)	98.8 (1.0)	99.2 (0.5)	98.4 (1.5)	53.4 (16.5)	100 (0)	100 (0)	100 (0)
MIRU-VNTRplus (†)	91.5 (1.3)	95.8 (3.2)	97.8 (1.8)	93.3 (1.4)	77.9 (5.7)	77.7 (13.4)	70.6 (24.1)	80.6 (17.9)	60.5 (32.0)
Same, no threshold (†)	97.3 (0.5)	97.9 (2.0)	99.6 (0.7)	98.8 (0.6)	90.5 (3.3)	80.6 (12.4)	70.6 (24.1)	100 (0)	100 (0)
TBminer - MIRU-VNTRplus	97.2 (0.6)	91.5 (4.2)	99.0 (0.9)	99.3 (0.5)	98.8 (1.3)	53.4 (16.5)	100 (0)	100 (0)	0 (0)
TBminer - Expert	97.1 (0.6)	95.1 (3.2)	99.1 (1.0)	99.2 (0.5)	97.4 (2.1)	46.8 (15.4)	100 (0)	100 (0)	0 (0)
TBminer - MIRU-VNTRplus (‡)	96.7 (0.7)	89.2 (4.3)	98.4 (1.3)	99.3 (0.5)	98.8 (1.3)	49.5 (16.1)	100 (0)	95.8 (8.2)	0 (0)
TBminer - Expert (‡)	96.5 (0.7)	89.6 (4.3)	98.5 (1.2)	99.0 (0.6)	98.4 (1.5)	43.2 (16.4)	100 (0)	100 (0)	0 (0)
RuleTB	94.6 (1.1)	92.2 (9.7)	86.3 (20.3)	88.7 (20.9)	92.8 (8.7)	85.9 (17.4)	89.8 (15.9)	89.0 (13.6)	92.3 (14.9)
StackTB	98.5 (0.5)	96.9 (2.8)	99.8 (0.6)	99.2 (0.5)	98.4 (1.6)	78.1 (13.6)	86.0 (24.7)	97.8 (6.9)	100 (0)

Table 2: Mean (standard deviation) accuracy of algorithms trained on refined lineages over 100 random partitions. Legend: ($\star$) = the MIRU-VTNRplus database is used for training; (†) = the training subset of the entire data is used for training; ‡= only the 15 most discriminative VNTR loci are used for the prediction. Note that TB-Insight is omitted because it does not identify lineages at this level of refinement.

Algorithm	Overall	LI	LII	LIIa	LIII	LIV	LV	LVI	LVII	LVIII	M. afri 1	M. afri 2	M. bovis
MIRU-VNTRplus (†)	84.7 (1.5)	77.5 (5.0)	97.7 (1.6)	93.5 (3.5)	87.0 (3.6)	71.8 (9.0)	68.5 (12.7)	85.7 (3.8)	73.2 (5.4)	67.6 (20.5)	78.5 (12.2)	67.0 (26.3)	81.2 (17.7)
Same, no threshold (†)	89.6 (1.3)	91.0 (3.9)	99.4 (0.7)	95.9 (3.0)	88.2 (3.3)	74.3 (8.5)	73.3 (10.9)	90.9 (3.1)	79.0 (5.1)	74.3 (17.9)	82.4 (11.0)	67.0 (26.3)	100 (0)
RuleTB	81.6 (1.8)	96.1 (2.7)	92.0 (3.4)	87.7 (5.2)	76.1 (5.6)	46.3 (9.7)	60.4 (16.9)	74.2 (4.3)	86.1 (6.3)	85.8 (18.6)	75.8 (15.6)	79.0 (30.3)	86.6 (12.7)
StackTB	92.7 (1.3)	98.7 (1.6)	99.6 (0.6)	95.7 (3.6)	91.6 (3.1)	85.4 (8.2)	75.1 (13.9)	93.4 (2.8)	80.2 (6.9)	87.3 (16.9)	79.3 (12.5)	82.5 (26.9)	98.0 (6.0)

Table 3: Mean (standard deviation) accuracy of algorithms trained on SNP clusters over 100 random partitions. Note that MIRU-VNTRplus (with the database as training set), TB-Insight and TBminer do not produce this classification, and are therefore omitted.

[1] Malioutov, D and Varshney, K. *Exact Rule Learning via Boolean Compressed Sensing* Proceedings of ICML-13 (2013), pp. 765-773.

[2] Friedl MA, Brodley CE (1997). *Decision tree classification of land cover from remotely sensed data*. Remote sensing of environment. 61(3):399-409.

[3] Rokach L, Maimon O (2002). *Top-down induction of decision trees classifiers-a survey*. IEEE transactions on systems, man and cybernetics: part c. 1(11):1-2.

[4] Friedman JH (2001). *Greedy function approximation: a gradient boosting machine*. Annals of statistics 1:1189-232.

[5] Ho TK (1998). *Nearest neighbors in random subspaces*. InJoint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), pp. 640-648. Springer Berlin Heidelberg.

[6] Aggarwal CC, Hinneburg A, Keim DA (2001). *On the surprising behavior of distance metrics in high dimensional space*. InInternational Conference on Database Theory, pp. 420-434. Springer Berlin Heidelberg.

[7] Greene WH (2012). *Econometric Analysis* Seventh edition. Boston: Pearson Education. pp. 803-806.

[8] Jaccard P (1901). *Étude comparative de la distribution florale dans une portion des Alpes et des Jura*. Bulletin de la Société Vaudoise des Sciences Naturelles. 37:547-579.