GENOTYPE PHENOTYPE MAPPING IN RNA VIRUSES - DISJUNCTIVE NORMAL FORM LEARNING

CHUANG WU, ANDREW S. WALSH and RONI ROSENFELD

School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA
*E-mail: chuangw@cs.cmu.edu, awalsh@cs.cmu.edu, *Roni.Rosenfeld@cs.cmu.edu*

RNA virus phenotypic changes often result from multiple alternative molecular mechanisms, where each mechanism involves changes to a small number of key residues. Accordingly, we propose to learn genotype-phenotype functions, using Disjunctive Normal Form (DNF) as the assumed functional form. In this study we develop DNF learning algorithms that attempt to construct predictors as Boolean combinations of covariates. We demonstrate the learning algorithm's consistency and efficiency on simulated sequences, and establish their biological relevance using a variety of real RNA virus datasets representing different viral phenotypes, including drug resistance, antigenicity, and pathogenicity. We compare our algorithms with previously published machine learning algorithms in terms of prediction quality: leave-one-out performance shows superior accuracy to other machine learning algorithms on the HIV drug resistance dataset and the UCIs promoter gene dataset. The algorithms are powerful in inferring the genotype-phenotype mapping from a moderate number of labeled sequences, as are typically produced in mutagenesis experiments. They can also greedily learn DNFs from large datasets. The Java implementation of our algorithms will be made publicly available.

1. INTRODUCTION

RNA viruses (including retroviruses), such as HIV, Influenza, Dengue and West Nile, impose a very significant disease burden throughout the world. Because of their very short generation time and low replication fidelity,[1] RNA viruses exhibit extensive variability at the nucleic acid and protein level which results in fast adaptation rate, and great ability to evade the immune system and antiviral drugs.[2,3] For example, HIV drug resistance has developed to all available drugs,[4] and some drug resistance mutations are probably present before the start of therapy;[5] Influenza resistance to Neuraminidase Inhibitor is rare but the resistance mutations are emerging and the resistance becoming more prevalent.[6]

Genotype-phenotype function learning is important first step in elucidating the mechanisms responsible for various viral phenotypes. It is also a crucial step towards inferring the phenotype from sequence alone, which has broad uses in clinical decision making (e.g. antiviral drug choice based on drug resistance) and in public policy (e.g. vaccine formulation based on immunogenicity and cross-reactivity).

We believe that by appropriately exploiting domain knowledge, computational methods can efficiently and correctly learn genotype-phenotype mapping. This can be combined with the large and rapidly growing sequence datasets to reduce the amount of required biological experimentation. The fast replicating RNA viruses provide us a large pool of RNA virus sequences data. There were 229,451 sequences in the HIV Sequence Database at Los Alamos National Laboratory by the end of 2007, an increase of 17% since the year before (http://www.hiv.lanl.gov/). This abundant data suggests a great opportunity for computational models to unveil the underlying mechanisms of phenotype changes. Specifically, by making best usage of the domain knowledge the models could capture the genotype-phenotype correlations and improve prediction performance. We believe that computational tools will be essential as exploratory and interpretation systems to support clinical decisions concerning the prediction of the phenotypes.[7]

RNA virus phenotypes typically result from multiple alternative mechanisms. Each mechanism is sufficient to explain the phenotypes and is constituted of a small number of key residues; yet each key residue alone may correlate weakly with the observed viral phenotype. This is biologically plausible and can be seen in a variety of biological evidences. For example, drug resistance is often a steric-structural problem, and the physical interactions with inhibitors involve more than one part of the target molecule, e.g. Protease Inhibitors (PIs) bind to four or more binding pockets in the protease substrate cleft of HIV viruses;[8] a variety of active site properties are playing roles

in the binding determination, such as residue types, hydrophobicity, charges, secondary structure, cavity volume, cavity depth and area etc. Therefore, multiple, alternative potential mechanisms exist. Each mechanism involves only a small number of mutations since it has to be "discovered" by the virus via random mutations. Thus overall only a small number of key residues are involved. Therefore, a short Disjunctive Normal Form (DNF, "OR" of "AND") would be an appropriate bias over the hypothesis space under these assumptions. DNF is a disjunction of conjunctions where every variable or its negation is represented once in each conjunction. DNFs of proteins provide a mapping between key residues and their phenotype, and are an informative abstraction of key residues for the construction. Another advantage of DNF is that it is a natural form of knowledge representation for humans to interpret.

The learning of DNFs is a machine learning technique to infer Boolean function relevant with a class of interest. It has been extensively used in electric circuit design, information retrieve,[9] chess game,[10] and so on. The learnability of DNFs has been a fundamental and hard problem in computational learning theory for more than two decades. Because of the combinatorial complexity, exhaustive search algorithms for finding solutions require huge computational resources. Our group has been developing algorithms for accelerating and optimizing the DNF learning for RNA virus phenotypes, based on biologically plausible assumptions. We are also concerned with the amount of data available and the learning efficiency of the algorithms. In this study, we develop fast exhaustive DNF learning algorithms under biologically plausible assumptions. The algorithms can learn DNFs either from only a few mutagenesis experiments or from large high-throughput datasets. The learning quality is evaluated by examining the biological interpretation and prediction quality of the functions on a variety of RNA virus datasets representing different phenotypes.

2. Related computational work

Existing work on computational and statistical inference of genotype-phenotype relationship focuses on population genetics, using linkage analysis and association studies. Linkage analysis is not applicable to our case because crossover is not a significant force in the evolution of most RNA viruses. Similarly, association studies are not applicable here because they can only detect single-locus associations, or else require exceedingly large datasets: for a typical scenario where up to a few dozen labeled sequences are available and the phenotype depends on 2-4 key residues that interact in a complex fashion, there is not enough power in statistical tests to identify these residues. More specifically, tests like those described in[11–13] look for association between each individual residue position and the phenotype. But if the phenotype is determined by a complex interaction among, say, four residue positions, then there will be only moderate association between any one of these positions and the phenotype label, and this association may not be reliably detected with the limited number of labeled sequences that are usually available. This is a weakness shared by all methods that look for phenotypic association with individual residue positions (call these "*position-specific association methods*"). This deficiency on the real data was described in.[14] Although position-specific association methods can be expanded to look for phenotypic associations with any pair or triplet of residues etc., the exponential growth in the number of covariates further reduces the power of the tests. An even more serious limitation of these methods is that they assume that the labeled data were independently sampled, a patently false assumption in most cases of interest.

Rule induction algorithms, such as simultaneous covering by decision tree algorithm,[15] and ordered list of classification rules induction[16] can also mine if-then rules, but they only discover small number of rules for efficient prediction or classification purposes. Sequence analyses using logic regression[17] and Monte Carlo Logic regression[18] adaptively identify weighted logic terms that are associated with phenotypes. These approaches do not explore the whole hypothesis space to identify all possible solutions; hence it is not guarantee to learn the global optimal solution.

Many state-of-the-art machine learning approaches have been applied to RNA virus genotype-phenotype mapping, such as support vector machine (SVM) regression,[19] decision tree classification,[20] statistical models,[21] neural network,[22,23] recursive partitioning,[24] linear stepwise regression,[25] support vector regression,[8] least-squares regression,[8] and least angle regression.[8] These models learn from a training data set and then test their performance using a test data set. The effort focuses on the prediction accuracy in a cross validation manner, but these approaches lack the intention to learn biologically meaningful and interpretable functions. Nonetheless, we will comprehensively

compare our DNF learning algorithms with these approaches on prediction quality.

We are not aware of any statistical or computational methods designed specifically to infer genotype-phenotype relationship in RNA viruses or other situations dominated by point mutations and small to moderate datasets.

3. Disjunctive Normal Form (DNF) learning algorithms

Disjunctive Normal Form (DNF) is a disjunction of conjunctions, where the conjunctions vary over positive and negative literals. Any given boolean function $f : \{0,1\}^d \rightarrow \{0,1\}$ can be written in an equivalent DNF. For example, a DNF formula is of the form:

$f(x_1, x_2, x_3) = x_1 \wedge x_3 + x_1 \wedge \neg x_2 \wedge x_3 + x_2$

where '$\wedge$' denotes 'AND', '+' denotes 'OR', '$\neg$' denotes negation, and 'x' is a binary literal. This example formula is a 2-term 3-DNF which contains two conjunctive terms (called clauses hereafter) with a maximum clause length of 3 literals. The size of the DNF formula is defined as the number of clauses it contains. A DNF formula represents a logic if-then rule, which is true only if the logic calculation of inputs is true. To adapt biological sequence data, the binary literals are extended to positional category variables. For example, an extended literal 'x = 5A' means 'x = Ind(the sequence item at the 5th position is 'A')', where 'Ind()' is an indicator function, and 'A' is the string representation of amino acid or nucleotide acids. This extension enables us to assign labels to any biological sequences. When a function assigns a positive label to an input sequence, we say that the function 'covers' the sequence. The goal of our DNF learning algorithm is to learn the shortest DNF(s) that cover all the positively labeled data, and do not cover any of the negatively labeled data.

Finding the minimum size DNF formula is a well-known NP-Complete problem;[26,27] hence there is no polynomial time learning algorithm. Existing practical solutions usually sacrifice completeness for efficiency. The existing heuristic or approximation approaches can be categorized into deterministic[9,28,29] and stochastic algorithms.[10,30] The deterministic methods include bottom-up schemes (learning clauses first and building DNFs in a greedy way) and top-down schemes (converting DNF learning to a Satisfiability problem). Stochastic methods randomly walk through the solution space to search for clauses but are not guaranteed to yield optimal solutions.

In this study, we aim to find the minimum size DNFs by making the assumption that only small numbers of key residues are involved in determining the functions. The assumption is biologically plausible and can be seen in a variety of RNA virus phenotypes:[14] **Drug resistance:** In Influenza, resistance to the M2 ion channel blockers amantadine and rimantadine is associated with two mutations in the M2 protein;[31] **Immunogenicity:** In HIV-1, decreased immunogenicity has been shown to be caused by three mutations in the gag protein;[32] **Pathogenicity:** In Avian Influenza, dramatically increased pathogenicity was found to be associated with a small number of mutations in the polyprotein cleavage site;[33] **Antigenicity:** In Influenza A, the investigation of the differences between the vaccine strain (A/Panama/2007/99) and the circulating (A/Fujian/411/02-like) virus showed that two mutations in the hemaglutinin protein are responsible for the antigenic drift.[34]

Using this assumption, our method:

(1) Converts DNF learning to learning k-DNF where $k \geq 1$ is the maximum size of conjunctive clauses. (Standalone DNF learning algorithm)
(2) Exhaustively learns monotone DNF(s) after feature selection (Monotone DNF learning algorithm)
(3) Greedily learns DNFs for hard problem settings.
(4) Extracts biologically meaningful solutions and better predicts phenotypes from genotypes.

3.1. *Standalone DNF learning*

Valiant[35] showed that for every constant $k \geq 1$, k term DNF can be PAC learned in polynomial time by k-CNF, i.e. CNFs with at most k literals in each clause. K-term DNF learning is essentially a combinatorial problem. The standalone DNF learning algorithm first learns a set of conjunctive

clauses deterministically with the maximum clause length of k (table 1), and then constructs DNFs from the clause pool. The construction process becomes a typical SET-COVER problem (table 2) after converting each clause into a set of sequences it covers. In response, the DNF learning algorithm is equivalent to finding the minimum number of sets that cover all the positive sequences. Although the SET-COVER problem is again NP-Complete, by limiting the maximum clause length, for typical RNA virus problem settings the number of clauses is usually manageable and the SET-COVER can be exhaustively completed. Typically, the number of possible clauses of size$k$ is up to L^k, where L is the sequence length. The actual number of clauses that appear in the dataset is much smaller than this number, especially for biologically conserved datasets. After equivalence filtering (see Section 3.4), given the datasets we evaluated, the number of learned clauses is usually about several hundreds.

The standalone algorithm can be extensively used to infer DNFs from small (a couple of sequences) to medium size (hundreds of sequences) datasets, or large conserved datasets.

Table 1. Clause learning algorithm

Clause Learning Algorithm (S, k):	
Input:	
	A set **S** of already-available labeled sequences
	k: assumed upper-bound length of clauses (a small positive integer)
Steps:	
	1. Enumerate all combination of literals to form conjunction clauses
	2. Record the set of (positive and negative) sequences that each clause covers (n_j^+, n_j^-)
Output:	
	The set of clauses C and the corresponding sequence index sets (N^+, N^-)

Table 2. DNF learning algorithm

Disjunctive Normal Form Learning Algorithm(C, n^+):	
Input:	
	A set **C** of clauses
	n^+: the set of positive sequence index to be covered by the clauses
Steps:	
	1. Euivalence filtering (see Section 3.4)
	2. Among the clauses that cover only the positive sequences, find a minimum set of clauses that cover all the positive sequences:
	2a. start from the clauses that cover the positive sequences which are rarely covered by other clauses
	2b. repeat 2a recursively until all the positive sequences n^+ are covered
Output:	
	The set of the shortest DNFs

3.2. *Monotone DNF learning after feature selection (MtDL)*

In machine learning, feature selection is a technique of selecting a subset of relevant features to build robust learning models or for prediction purposes. Here we use feature selection to choose the set of features that we believe the solution DNFs are based on, and then construct DNFs within the selected feature space. Note that the feature selection is only used to narrow the feature space, but does not infer any mapping functions. The monotone DNF learning algorithm then exhaustively builds DNFs based on the selected features. By doing this, the size of the solution space is greatly narrowed, as a result MtDL does not need to limit the maximum length of clauses and can search the hypothesis space more thoroughly than the standalone DNF learning algorithm.

The Monotone DNF learning algorithm after feature selection (MtDL) is explained in table 5. The learning algorithm first enumerates all possible literals within the selected features, and then combines them into conjunctive clauses. MtDL differs from the standalone algorithm in that it does not limit the maximum size of clauses but completely considers all possible combinations. Take a typical example: L features are selected after feature selection, where L is usually much smaller than the sequence length. In step 1 there should be at most $M = 20 * L$ possible literals in the case of protein sequences. Because the literals from the same position will not appear in the same conjunctive clause, we do not need to consider all 2^M combinations, and instead only combinatorially choose up to L literals from M. Hence the total number of clauses is at most $N = (M \text{ choose } L)$. In reality, depending on the divergence and the amount of the data, the actually number of possible literals is always much smaller than this number. Furthermore, in step 4, the N clauses will be pre-filtered by removing the clauses that cover any negative sequences. When the clause pool is ready, in step 5 the algorithm incrementally constructs the combination of clauses to be candidate DNFs and examines the coverage of sequences. MtDL starts from 1 clause, and checks the next larger number if no solution is found. The algorithm terminates when the DNFs cover all positive sequences but not any of the negative sequences. In the result section we show that in practice MtDL runs fast on real RNA virus datasets.

Feature selector: The choice of feature selector is critical to the MtDL algorithm. The best feature selector for MtDL needs to guarantee that the selected feature space is a superset of the DNF solution space, and as small as possible for fast calculation. The Combinatorial Filtering (CF) algorithm we developed[14] works seamlessly with MtDL as a feature selector. CF() efficiently identifies the smallest set of positions that completely explains the differences between classes, thus MtDL can definitely learn DNFs based on these positions. In the following evaluation, MtDL always runs together with CF(). Notwithstanding, other feature selection methods, such as LASSO, Logistic Regression with regularization, or dimension reduction methods like PCA, are also good candidate selectors, but in these cases, the coverage threshold might need to be set (section 3.5).

Table 3. Monotone DNF learning algorithm

Monotone DNF Learner (F, S):	
Input:	
	F: A set of selected features (by CF(), for example)
	S: the labeled training datasets
Steps:	
	1. Construct {L}, the list of literals in the features (e.g. 5A).
	2. Throw out L that does not cover any positive sequences.
	3. Combinatorial construct {Clauses}, the list of conjunctive clauses from {L}, (e.g. $5A \wedge 8C$). The possible combinations are $\|L\|$ chooses $1, 2, .., \|F\|$.
	4. Throw out the conjunctive clauses that cover any negative sequences.
	5. Incrementally construct {DNF}, the list of disjunctive normal form that covers all positive sequences but no negative sequences: starts from 1 clause, construct DNFs from {Clauses}, try the next larger number if no solution learned.
Output:	
	The set of the shortest DNFs

3.3. *Greedy versions of both algorithms*

As we will show in the result section, with typical RNA virus datasets, the standalone and monotone DNF learning algorithms learn DNFs efficiently. In cases of very large dataset, both algorithms are modified to greedy versions to learn DNFs rapidly. The greedy versions only differ from the exhaustive versions in the DNF construction step. Instead of exhaustively combining all clauses to construct DNFs, the greedy algorithms iteratively select the clause that covers the largest number of the uncovered positive sequences until all the positive sequences are covered.

3.4. *Equivalence filtering*

Computationally equivalent clauses cover the same set of sequences while differing in their composition literals. They are equivalent in DNF functions in the sense that replacing one clause with its equivalent clauses will not change the predictions of the DNF on the same training set. Equivalent clauses are very common in RNA virus datasets; therefore, during DNF learning process equivalent clauses are filtered and only one of them is used as the representative to construct DNFs. By using equivalence filtering the DNF learning running time is greatly reduced. Note that the equivalence filtering is only for computational efficiency purpose. After learning DNFs, all clauses that have equivalent clauses will be expanded to recover all the DNFs.

3.5. *Avoiding over-fitting and robustness to noise*

The following two techniques are used to avoid over-fitting and make the algorithms robust to noise:

(1) **DNF pruning**: similar to the pruning of decision tree, after learning DNFs the clauses that only cover a small number of sequences may be pruned if removing them results in an increase of the prediction accuracy on the test dataset. The advantages of pruning are:
   - Avoiding over-fitting, because irrelevant clauses are removed.
   - The DNFs are shorter, which makes them easier to understand and more biologically meaningful.
   - Robust to noise, because pruned DNFs ignore the clauses/literals rendered meaningless by noise.

(2) **Threshold setting**: set thresholds of the fractions of the sequences that the learned DNF(s) cover. The DNF learning algorithms can be easily modified to terminate when at least a fraction p of the positive sequences are covered, and at most a fraction n of the negative sequences can be covered by the DNFs. The thresholds p and n are determined in a cross-validation way. Similar to pruning, the threshold setting method can also avoid over-fitting, learn shorter DNFs and be robust to noise. One advantage of threshold setting over pruning method is that threshold setting usually achieves better prediction quality.

3.6. *Extension of literals*

The literals can also be extended to negation of one amino acid or a subset of the amino acids.

3.7. *Extension to multiple class data*

The algorithm is applicable to multiple class data by running multiple times with each time one of the classes is made positive class and the rest are merged as negative class.

4. RESULTS

4.1. *Demonstrating DNF learning algorithms consistency*

The DNF learning algorithms will first be validated on simulated protein sequences with hypothetical target functions. We will use this stage to validate the algorithms' consistency. When generating simulated sequences, we match the position-specific amino acid distributions to those of a real protein datasets, and then generate random phenotypic target functions (making sure they did not label the entire dataset with the same value). We use 732 HIV-1 gp160 protein sequences (downloaded from LANL), and assumed a variety of target functions (e.g. $(70a \wedge 9l \wedge 11p \wedge 70t) + (62l \wedge 45m \wedge 36y \wedge 9l) + (62P \wedge 53V \wedge 36s) + (83i \wedge 45I) = +)$. Each target function contains a number of clauses, and it is used to label the sequences accordingly. Notice that this method enables us to generate as many sequences as we want so we can test the algorithm convergence under a variety of conditions. We repeated this process many times, and in all of these cases both standalone algorithm and MtDL algorithm converged to the target functions with moderate number of sequences.

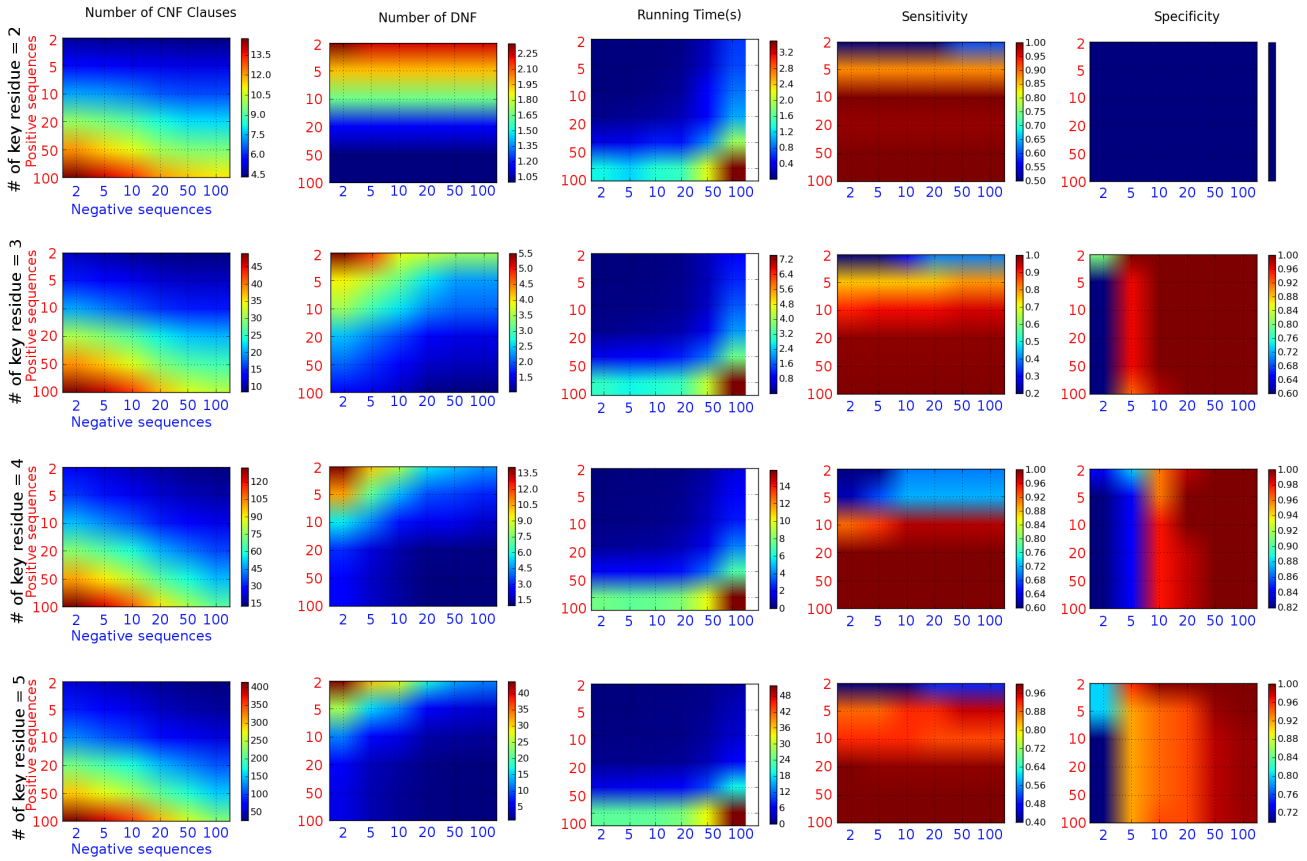

Fig. 1. **The evaluation of MtDL algorithm on simulated sequences.** From left to right, the number of CNF clauses, the number of DNFs, running time, prediction sensitivity and specificity are plotted as functions against the number of key residues assumed in the target function (rows), and the number of positive sequences and negative sequences (vertical columns and horizontal rows of small colored squares). The numerical values of the colors are shown in the colorbar. Take the top left chart for example, when the key residues are assumed to be 2 in the target function, with say 100 positive and 2 negative sequences used, the number of CNF clauses is about 12 (red color means higher value as indicated in the colorbar).

4.2. *Measuring inference efficiency (convergence rate as function of dataset size)*

To assess the efficiency of our learning method, we would like to understand the relationship between the complexity of the function to be learned and the number of training examples needed to converge to it. Namely, we would like to know the convergence rate as a function of the amount and type of available sequences and the complexity of the genotype-phenotype mapping. This is important because the available number of sequences vary for different datasets. Based on the convergence rate we can assess the likelihood of convergence, and whether (and how much) further experimentation will be needed.

To do this, we used 588 aligned sequences of HIV protease protein downloaded from the Stanford HIV database, with an aligned length of 99. We then randomly generated putative binary target functions, each depending on a small number (2..5) of literals. For each such target function, the 588 sequences were labeled accordingly. The DNF learning algorithm was then run 20 times, each time assuming a different target function to produce a statistically robust result. As an illustration, we will show the evaluation results of MtDL in the following sections, and the simulation result of the standalone algorithm is similar.

Convergence Rate: As described in the introduction, we are concerned with the relation between the amount of available data and the convergence of the algorithm. It is therefore most meaningful to compare the convergence of the hypothesis space under our algorithm with this method. Figure 1 shows the number of DNFs learned by the algorithm as a function of the number of literals in the target function (# of key residues, top-to-bottom), and the number of positive and negative sequences (vertical and horizontal rows of small colored squares, respectively, with values of 2, 5, 10, 20, 50, 100 sequences each). Blue color indicates convergence (i.e. one single DNF is learned given the amount of data, and the learned DNF is exactly the same as the target function), and red color indicates alternative DNFs exist. The number of DNFs reduces with more available sequences, and in all of the cases, MtDL algorithm converges with only about 50 positively labeled and 50 negatively labeled sequences.

Another important factor in measuring efficiency is the running time due to the combinatorial nature of DNF learning. Recall that in MtDL, the number of candidate clauses in step 4 is bounded by 2^L and the number of DNFs is bounded by 2^{2^L}, where L is the number of literals. L increases when more sequences are available, and this explains why, in the "running time" column the red color, which indicates the longest running time, is at the right bottom corner. Note that the longest running time is still on the order of seconds in the simulation.

The prediction sensitivity and specificity showed that the algorithms converge with only moderate numbers of sequences. Interestingly, the prediction quality chars are symmetric in that if we flip the labels of the data, the prediction accuracy will remain the same. This is important because although our DNF learning algorithms identify DNFs that only cover the whole positive space, the sequences in both classes contribute equally to the learning.

4.3. *Retrospectively validating DNF learning algorithms when ground truth is known*

We retrospectively validated the MtDL algorithm by testing it on datasets with real viral protein sequences where the genotype-phenotype mapping is already known and assumed to be correct. We compiled a number of datasets covering several RNA viruses with varying degree of average sequence identity (SI) and a variety of phenotypes, including Avian Flu High/Low pathogenicity (4 mutations in HA proteins changed the pathogenicity from low to high in H5N2 Influenza HA, SI: 95%),[33] Influenza H3N2 antigenicity shift (2 mutations in HA shifted the antigenicity of Influenza H3N2, SI: 93%),[34] SIV Env neutralizability (2 mutations in SIV Env proteins determined the neutralizability of SIV, SI: 99%),[36] FIV tropism in CRFG cells (2 mutations in FIV polymerase PA subunit made it unable to replicate in CRFK cells, SI: 95%).[37] These conclusions were made from mutagenesis experiments that were chosen empirically or by domain knowledge. We applied our MtDL+CF (using CF as the feature selector) algorithm on the same set of mutagenesis sequences to predict the key residues for the phenotype changes. For all the tests performed, our algorithm converged to the correct answer(s). In contrast, the conventional position-specific association method we selected as comparison,[11] can only predict positions of importance, but our MtDL+CF algorithm explicitly learns the actually mapping functions. Even so when only comparing the positions identified, the conventional method only correctly identifies the positions in one of the datasets, and yields high false positive and false negative rates in the other four datasets (Table 4). This demonstrates our arguments in section 2 that if the phenotype is determined by a complex interaction, the traditional methods cannot detect all the key residues correctly.

4.4. *The utility of DNF learning algorithms on large datasets*

To demonstrate the applicability of MtDL to learn biologically meaningful results, MtDL+CF was applied to a large, divergent dataset, the HIV drug resistance dataset (download from Stanford HIV database). The dataset was retrieved from the Stanford HIV database, including seven Protease Inhibitor drugs and eleven Reverse Transcriptase Inhibitor drugs. We ran the MtDL algorithm on all the drug datasets and learned very short and interpretable DNFs (Table 5). For example, a protein is resistant to NFV when position 9 is I or (position 63 is I and position 9 is not I and

Table 4. **Comparing DNF learning with position-specific association methods** Retrospective comparisons of the DNF learning algorithm on a variety of datasets

Data set name (# pos/# neg seq)	**Golden standard (identified mutations)**	**DNF(s) learned by MtDL**	**Positions identified by Traditional method**
H3N2 hema Antigenicity shift (490pos/421neg)	145H, 146Q	145H+146Q	18, 67, 122, 145, 146
H5N2 hema Pathogenicity (4pos/11neg)	275K, 275T, 323K, 324R, 325K	323K+324R+325K	275, 323, 324, 325
FIV tropism (3pos/7neg)	30E, 32K	30K∧32E	32
SIV Envelope Neutralizability (8pos/5neg)	179N, 337R	179N+337R	331, 348

position 87 is not N) .

The purpose of these concise DNFs is three folds: 1) to identify the key positional determinants of drug resistance, e.g. position 89I for RTV, position 36 and 45 for APV,etc.. These positions have been identified by experiments and reported in literatures; 2) quantitatively describe how the residues in these positions combine to produce resistance; and 3) proposed new interpretation of HIV drug resistance mechanisms that have potential to be validated by domain experts.

Table 5. DNFs learned from HIV drug resistance dataset

Drug type	Drug	DNF = sensitivity/specificity; in the DNFs, lowercases mean negation
PI	NFV	$9l + (63l \wedge 9i \wedge 87n) = 0.902/0.834$
	RTV	$(81i \wedge 81v) + 83i + (70a \wedge 70l \wedge 89l \wedge 70t) = 0.981/0.988$
	LPV	$(9l \wedge 53i) + (9l \wedge 45m \wedge 9h \wedge 9m) = 0.961/0.965$
	APV	$(36s \wedge 45m \wedge 36y \wedge 9l) = 0.787/0.961$
	IDV	$(70a \wedge 9l \wedge 11p \wedge 70t) + (62l \wedge 45m \wedge 36y \wedge 9l) + (62P \wedge 53V \wedge 36s) + (83i \wedge 45I) = 0.965/0.982$
	SQV	$(9r \wedge 83i) + (62q \wedge 53i \wedge 89l \wedge 70l) + (45i \wedge 89l \wedge 70t \wedge 70a) + (76V \wedge 89l \wedge 81v \wedge 9l) = 0.899/0.963$
	ATV	$(70a \wedge 9l) + (89l \wedge 76V \wedge 81a) = 0.833/0.910$
NRTI	DDI	$150M + (68s \wedge 68t \wedge 68d \wedge 68n) + (74v \wedge 42n \wedge 74t) = 0.738/0.985$
	AZT	$(73v \wedge 34 - \wedge 214t \wedge 214d) = 0.848/0.988$
	D4T	$(209l \wedge 214d \wedge 34t \wedge 214t) + (68t \wedge 34m \wedge 214d \wedge 214t) + (68T \wedge 117I \wedge 66N) = 0.797/0.956$
	TDF	$(34i \wedge 66N \wedge 214t \wedge 183v) + (68g \wedge 19r \wedge 214Y \wedge 68t) + (34V \wedge 68t \wedge 214f \wedge 214t) = 0.784/0.980$
	ABC	$183V + (214d \wedge 121p \wedge 209l \wedge 82k) + (82k \wedge 66d \wedge 214Y \wedge 180c) = 0.940/0.944$
NNRTI	NVP	$(102k \wedge 102r) + (189g \wedge 102n) + (180y \wedge 100e) = 0.868/1.0$
	DLV	$(210t \wedge 102N) + (180y \wedge 100q \wedge 226F \wedge 210t) = 0.915/0.994$
	EFV	$(102r \wedge 102k) + (102s \wedge 189g) = 0.871/0.997$

4.5. *Improved prediction performance of DNF learning algorithms on the HIV drug resistance problem*

To demonstrate the prediction power of our DNF learning algorithms, we examine the prediction quality of both standalone and MtDL learning algorithms on two well-known Biology datasets respectively: the HIV drug resistance dataset and the UCI promoter gene dataset (details in section 4.6). Many state-of-the-art machine learning models, such as Support Vector Machine, Decision Trees, Neural Networks, Nave Bayes etc, have been tested on the datasets to learn genotype-phenotype mapping, and the five-fold cross-validation prediction quality was reported.[8] In the HIV drug resistance dataset, drug resistance levels are defined as fold-increased resistance compared to the wild type virus strain; therefore, for classification models the numerical resistance values were converted to multiple class labels by setting resistance thresholds. We use the thresholds suggested on the website, and select the sequences with significant resistant values and susceptible values while ignoring those with weak resistance or weak susceptible labels. The five-fold cross-validation prediction accuracies of Protease Inhibitor are shown in table 6. The standalone DNF learning algorithm outperforms other machine learning algorithms in 4 out of the 7 PI datasets (Table 6). The result suggests that by exploiting domain-knowledge to reduce the running time, the exhaustive

Table 6. Comparing Standalone DNF learning with published machine learning algorithms on HIV Protease Inhibitor datasets. The numbers of positively labeled and negatively labeled sequences in the datasets are shown, as well as and the prediction accuracies of 1) Standalone DNF, 2) Z-score,[11] 3) Naive Bayes (from Weka), 4) SVM (svm light software, default parameters), 5) Decision Tree (Weka, ID3 algorithm), 6) Winnow (Weka). The highest accuracy of each drug is highlighted in bold

Prediction accuracy (%)	NFV	SQV	IDV	RTV	APV	LPV	ATV
#pos/#neg sequences	194/211	119/321	115/279	154/244	47/308	103/142	42/111
Standalone DNF	93.5	**91.8**	**91.7**	96.1	**96.1**	88.2	**93.3**
Z-score	74.6	87.3	**91.7**	87.4	92.3	90.5	88.8
NaiveBayes	**95.1**	75.1	78.4	93.2	87.3	92.7	73.1
SVM (svm light)	77.2	74.2	83.4	92.2	87.5	86.3	72.6
DT	94.0	89.0	90.1	**98.6**	91.8	**98.6**	78.5
Winnow	91.1	84.7	89.9	94.6	91.1	94.6	85.9

Table 7. Comparing MtDL+CF with published machine learning algorithms on Promoter Gene dataset

System	Errors	Comments
MtDL + CF	4/106	No domain knowledge required
KBANN	4/106	A hybrid ML system that uses domain knowledge to initial the network structure
BP	8/106	Std backprop with one hidden layer
O'Neill	12/106	Ad hoc technique from the bio. lit.
Nearest neighbor	13/106	k-nearest neighbor, $k = 3$.
ID3	19/106	Quinlans decision-tree builder

algorithms achieve better prediction performance, and DNF turns out to be a reasonable bias on the hypothesis space as genotype-phenotype mapping functions for HIV drug resistance.

4.6. *Improved prediction performance on the UCI Promoter Gene dataset*

Another dataset we use to evaluate our DNF learning algorithms' prediction power is the popular UCI's promoter gene dataset, which has been studied with many machine learning models. The task is to predict promoters from DNA sequences of nucleotides, A, C, G, or T. The dataset contains 53 promoter sequences and 53 non-promoter DNA sequences. In Biology, the promoters are characterized by special motifs at certain positions from the transcription starting location, e.g. "cttgac" motif at +37 position indicates a promoter region. However, deriving all such domain theories is impractical and not meaningful. Computationally machine learning algorithms showed promising prediction performance on this dataset (Table 7). Among them the knowledge-based artificial neural network (KBANN)[38] achieves the best accuracy of 4 out of 106 errors in a held-out test manner.

The KBANN model is a hybrid system of both Explanation-based learning (EBL)(a system that corporate pre-existing knowledge) and Empirical learning system (learning solely from training examples). In[38] they argue that the hybrid system should be superior, in terms of classification accuracy, to empirical learning systems. On the Promoter Gene dataset, KBANN learns a neural network model and translates a set of domain theories to initial the neural network structure. The error rate is the number of wrongly predicted examples in a leave-one-out cross-validation (LOOCV) manner. Three other machine learning algorithms, standard back propagation, Quinlan's ID3, O'Neill's ad hoc partial pattern matching, and the "nearest neighbor" are compared in Table 7.

We employed the same LOOCV method on MtDL algorithm, and selected CF() as the feature selector. Although the prediction performance of MtDL+CF is the same as the best one KBANN, MtDL+CF does not require any pre-existing domain knowledge, which is not always available.

5. CONCLUSIONS

We developed two efficient DNF learning algorithms under the assumption that DNF is an appropriate bias over the hypothesis space for RNA virus phenotype datasets. The assumption is biologically

plausible and very important to our algorithms, because it reduces the hypothesis space greatly to make the computational hard problem solvable. We also demonstrated the learning efficiency and consistency on simulated sequences, showed the strength of the methods in learning biological meaningful mapping functions and showed superior prediction accuracies to positional-specific methods and other machine learning methods.

We aimed to learn the minimum size DNFs even though the exact learning is NP-complete. Compared to existing heuristic algorithms that only focus on learning time and learnability, we exploit the domain knowledge and develop efficient exhaustive algorithms to learn the shortest DNFs. We also applied a number of techniques to accelerate the DNF learning process, including setting the maximum length of clauses in standalone algorithm, using feature selector (CF) in MtDL to narrow down the searching space, equivalence filtering of the clauses, and extending both algorithms to greedy versions. This enables the algorithms to run over very large datasets. Notwithstanding, as shown in the result section, the DNF learning algorithms are also powerful in extracting DNFs from only a small numbers of sequences.

We focus on the learning from mutagenesis data where the data is highly reliable and the alignment is well defined. In the cases of noisy data and low quality alignment when combinatorial algorithms usually suffer more than statistical models, we use pruning and thresholds to make the algorithms robust to noise.

Our goal in this work has been to aid biological investigation by learning the genotype-phenotype mapping. Since this is our focus, we compared our method to other methods designed to do the same. Our algorithms explicitly learn genotype-phenotype mappings that are interpretable to humans, so that the mapping functions can not only predict phenotypes from genotypes along, but also unveil biologically meaningful explanations. The algorithms can learn DNFs from different sizes of data: ranging from a few sequences to large high-throughput datasets, and show superior prediction performances. In contrast, given the limited data, the positional-specific association methods would be ineffective if they were to be applied to the full set of protein positions because there is not enough statistical power for the inference. Given full size of dataset, our DNF learning algorithms outperformed other published machine learning algorithms on two common datasets.

We successfully demonstrated the learning efficiency and the prediction power of our DNF learning algorithms on RNA virus datasets, and the algorithms can be extensively used on other domains where similar assumptions hold.

6. ACKNOWLEDGEMENTS

We are grateful to Dr. Elodie Ghedin, Dr. Ted Ross and Dr. Kelly Cole for useful discussions.

References

1. J. W. Drake and J. J. Holland, *PNAS* **96**, 13910 (1999).
2. D. W. Klein, L. M. Prescott and J. Harley, *Microbiology* (Dubuque, 1993).
3. E. Domingo and J. Holland, *Annu. Rev. Microbiol* **51**, 151 (1997).
4. D. Pillay and M. Zambon, *BMJ* **317**, 660 (3 1998).
5. R. W. Shafer, *http://hivinsite.ucsf.edu/* (2004).
6. M.-A. Rameix-Welti, V. Enouf, F. Cuvelier, P. Jeannin and S. van der Werf, *PLoS Pathogens* **4**, 1 (2008).
7. A. Carvajal-Rdriguez, *Recent Patents on DNA and Gene Sequences* **1**, 63 (2 2007).
8. S.-Y. Rhee, J. Taylor, G. Wadhera, A. Ben-Hur, D. L. Brutlag and R. W. Shafer, *PNAS* **10**, 53 (2006).
9. S. N. Sanchez, E. Triantaphyllou, J. Chen and T. Liao, *Computer & Operations Research* **29**, 1677 (2002).
10. U. Ruckert and S. Kramer, 648 (2003).
11. S. S. Hanenhalli and R. B. Russell, *JMB* **303**, 61 (2000).
12. L. A. Mirny and M. S. Gelfand, *JMB* **321**, 7 (2002).
13. F. Pazos, A. Rausell and A. Valencia, *Bioinformatics* **22**, 1440 (2006).
14. C. Wu, A. Walsh and R. Rosenfeld, *IEEE-BIBE* (2010).
15. J. R. Quinlan, Induction of decision trees, in *Readings in Machine Learning*, eds. J. W. Shavlik and T. G. Dietterich (Morgan Kaufmann, 1990) Originally published in.
16. P. Clark and R. Boswell, *Proceedings of the Fifth European Working Session on Learning* **482**, 151 (1991).
17. C. Kooperberg, I. Ruczinski, M. L. LeBlanc and L. Hsu, *Genetic Epidemiology* **21**, 626 (2001).

18. C. Kooperberg and I. Ruczinski, *Genetic Epidemiology* **28**, 157 (2005).
19. N. Beerenwinkel, T. Lengauer, J. Selbig, B. Schmidt, H. Walter, K. Korn, R. Kaiser and D. Hoffmann, *IEEE* **16**, 35 (11 2001).
20. N. Beerenwinkel, B. Schmidt, H. Walter, R. Kaiser, T. Lengauer, D. Hoffmann, K. Korn and J. Selbig, *PNAS* **99**, 8271 (6 2002).
21. G. DiRienzo1, V. DeGruttola1, B. Larder and K. Hertogs, *Statistics in Medicine* **22**, 2785 (2003).
22. S. Draghici and R. B. Potter, *Bioinformatics* **19**, 98 (1 2003).
23. D. Wang and B. Larder, *The Journal of Infectious Diseases* **188**, 653 (3 2003).
24. A. D. Sevin, V. DeGruttola, M. Nijhuis, J. M. Schapiro, A. S. Foulkes, M. F. Para and C. A. B. Bouche, *The Journal of Infectious Diseases* **182**, 59 (2000).
25. K. Wang, E. Jenwitheesuk, R. Samudrala and J. E. Mittler, *Antiviral Therapy* **9**, 343 (2004).
26. Brayton (1985).
27. Gimpel (1965).
28. E. Triantaphyllou and A. L. Soyster, *Computers & Operations Research* **23**, 783 (1999).
29. Kamath, *Mathematical Programming* **57**, 215 (2005).
30. U. Ruckert, S. Kramer and L. D. Raedt, 405 (2002).
31. R. B. Belshe, M. H. Smith, C. B. Hall, R. Betts, and A. J. Hay, *Journal of virology* **62**, 1508 (5 1988).
32. S. Andre, B. Seed, J. Eberle, W. Schraut, N. Bultmann and J. Haas, *Journal of Virology* **74**, 2628 (2000).
33. M. Garcia, J. M. Crawford, J. W. Latimer, E. Rivera-Cruz and M. L. Perdue, *Journal of General Virology* **77**, 1493 (1996).
34. H. Jin, H. Zhou, H. Liu, W. Chan, L. Adhikary, K. Mahmood, M. Lee and G. Kemble, *Viology* **336**, 113 (2005).
35. L. G. Valiant, *Proceedings of the sixteenth annual ACM symposium on Theory of computing table of contents* , 436 (1984).
36. K. Cole and T. Ross, *Current HIV Res.* **5**, 505 (2007).
37. E. J. Verschoor, L. A. Boven, H. Blaak, A. Vliet, M. C. Horzinek, and A. Ronde, *American Society for Microbiology* **69**, 4752 (8 1995).
38. G. G. Towell, J. W. Shavlik and M. O. Noordewier, *AAAI-90 Proceedings* (1991).