Predicting virus mutations through relational learning

Elisa Cilia[1], Stefano Teso[2], Sergio Ammendola[3], Tom Lenaerts[1,4], Andrea Passerini*[2]

[1] MLG, Département d'Informatique, Université Livre de Bruxelles, Boulevard du Thriomphe - CP 212, 1050 - Brussels, Belgium
[2] Department of Computer Science and Information Engineering, University of Trento, via Sommarive 14, I-38123 (Povo) Trento, Italy
[3] Ambiotec sas, R&D, Via Appia Nord 47, 00142 Cisterna di Latina (LT), Italy
[4] AI-lab, Vakgroep Computerwetenschappen, Vrije Universiteit Brussel, Pleinlaan 2, 1050 - Brussels, Belgium

Email: Elisa Cilia - ecilia@ulb.ac.be; Stefano Teso - teso@disi.unitn.it; Sergio Ammendola - ricercaesviluppo@ambiotec.it; Tom Lenaerts - tlenaert@ulb.ac.be; Andrea Passerini*- passerini@disi.unitn.it;

*Corresponding author

Abstract

Background: Viruses are typically characterized by high mutation rates, which allow them to quickly develop drug-resistant mutations. Mining relevant rules from mutation data can be extremely useful to understand the virus adaptation mechanism and to design drugs that effectively counter potentially resistant mutants.

Results: We propose a simple relational learning approach for mutant prediction where the input consists of mutation data with drug-resistance information, either as sets of mutations conferring resistance to a certain drug, or as sets of mutants with information on their susceptibility to the drug. The algorithm learns a set of relational rules characterizing drug-resistance and use them to generate a set of potentially resistant mutants.

Conclusions: Promising results were obtained in generating resistant mutations for both nucleoside and non-nucleoside HIV reverse transcriptase inhibitors. The approach can be generalized quite easily to learning mutants characterized by more complex rules correlating multiple mutations.

Background

HIV is a pandemic cause of lethal pathologies in more than 33 million people. Its horizontal transmission trough mucosae is difficult to control and treat because the virus has a high virulence and it infects several type of immune surveillance cells, such as those characterized by CD4 receptor (CD4+ cells). The major problem in treating the human virus infection is the drug selectivity since the virus penetrates in the cell where it releases its genetic material to replicate itself by using the cell mechanisms. A drug target is the replicating apparatus of the cell. HIV antiviral molecules will be directed against several cells such as macrophages or lymphocytes T to interfere with viral replication. The HIV releases a single-strand RNA particle, a reverse transcriptase and an integrase into the cell cytoplasm. Quickly the RNA molecule is retro-transcribed in a DNA double strand molecule, which is integrated into the host genome. The integration events induce a cellular response, which begins with the transcription of the Tat gene by the RNA polymerase II. Tat is a well-known protein responsible for the HIV activation since it recruits some cytoplasm host proteins involved in the expression of viral genes. Remarkably, HIV can establish a life-long latent infection by suppressing its transcription, thus making ineffective the large part of antiviral drugs aimed at controlling the viral replication. However replicating viruses adopt several drug resistance strategies, for instance, HIV induces amino acid mutations reducing the efficacy of the pharmaceutical compounds. The present work is aimed at gaining knowledge on mutations that may occur into the viral RNA transcriptase [9]. This is an important target to develop antiretroviral medicines and different types of molecules have been found active: the Nucleoside Reverse Transcriptase Inhibitors (NRTI) and Non NRTI (NNRTI). Although RNA RT inhibitors are active the HIV virus quickly, more than frequently, changes the RNA RT encoding sequence thus acquiring drug resistance. The antiviral therapy is based on the use of cocktails of molecules including new RNA RT inhibitors, thus a computational approach to predict possible mutation sites, and their sensibility to drug is an important tool in drug discovery for the antiretroviral therapy.

Computational methods can assist here by exploring the space of potential virus mutants, providing potential avenues for anticipatory drugs [1]. To achieve such a goal, one first needs to understand what kind of mutants may lead to resistance.

A general engineering technique for building artificial mutants is referred to as *rational design* [2]. The technique consists in modifying existing proteins by site directed mutagenesis. It relies on a deep domain knowledge in order to identify candidate mutations that may affect protein structure or function. The process typically involves extensive trial-and-error experiments and is also aimed at improving the understanding mechanisms of a protein behavior.

In this work we report on our initial attempt to develop an artificial system mimicking the rational design process. An Inductive Logic Programming (ILP) learner [3] is trained to extract general rules describing mutations relevant to a certain behavior, i.e. resistance towards certain inhibitors. The learned rules are then used to infer novel mutations which may induce similar resistance.

We consider here two learning settings: in the first one we rely on a training set made of single amino acid mutations known to confer resistance to a certain class of inhibitors (we will refer to this as mutation-based learning and preliminary promising results in this setting were described in [4]), in the second we learn the rules directly from mutants (comprising of 1 to n amino acid mutations) that have been experimentally tested for their resistance to the same classes of inhibitors (we will refer to this as mutant-based learning). This second setting is actually the most common situation, in which one is presented with a number of mutants together with some evidence of their susceptibility to certain treatments. The goal is then to extract rules and potential mutations explaining why certain mutants confer resistance and which other ones produce the same effect.

Many machine learning methods have been applied in the past to mutation data for predicting single amino acid mutations on protein stability changes [5] and the effect of mutations on the protein function [6, 7] or drug susceptibility [8]. To the best of our knowledge this is the first attempt to learn relational features of mutations affecting a protein behavior and use them for generating novel relevant mutations. Furthermore, even if we focus on single amino acid mutations in our experimental evaluation, our approach can be straightforwardly extended to multiple point mutations, and we are actively working in this direction. Note that the other approaches first generate all potential mutations and then decide which of them leads to resistance, whereas we produce the relevant ones directly.

We report an experimental evaluation focused on HIV RT. RT is a well-studied protein: a large number of mutants have been shown to resist to one or more drugs and databases exist that collect those data from different sources and make them available for further analyses [10].

We tested the ability of our approach to generate drug-resistant amino acid mutations for NRTI and NNRTI. Our results show statistically significant improvements for both drug classes over the baseline results obtained through a random generator. Mutant-based results confirm our previous findings [4] extending them to the more natural setting in which only information on mutant behavior is observed. Additional background knowledge allows to produce here more compact and simple hypotheses, which have the advantage of generating a reduced number of candidate mutations.

The approach can be in general applied in mutation studies aimed at understanding protein function. By searching for residues most likely to have a functional role in an active site, the approach can for instance be used in the engineering of enzyme mutants with an improved activity for a certain substrate.

Results

Datasets

We applied our approach to predict HIV RT mutations conferring resistance to two classes of inhibitors: Nucleoside RT Inhibitors (NRTI) and Non-Nucleoside RT Inhibitors (NNRTI). The two classes of inhibitors differ in the targeted sites and rely on quite different mechanisms [11, 12]. NNRTI inhibit the reverse transcriptase by binding to the enzyme active site, therefore directly interfering with the enzyme function. NRTI are instead incorporated into the newly synthesized viral DNA for preventing its elongation.

We collected datasets for both mutation-based and mutant-based learning. The former (Dataset 1) is a dataset of amino acid mutations derived from the Los Alamos National Laboratories (LANL) HIV resistance database[1] by Richter et al. [13], who used it to mine relational rules among mutations. It consists of 95 amino acid mutations labeled as resistant to NRTI and 56 labeled as resistant to NNRTI, over a set of 581 observed mutations. For the mutant-based setting, we collected (Dataset 2) HIV RT mutation data from the Stanford University HIV Drug Resistance. The database provides a dataset of selected mutants of HIV RT with results of susceptibility studies to various drugs, and was previously employed [8] for predicting drug resistance of novel (given) mutants[2]. It is composed of 838 different mutants annotated with susceptibility levels (low, medium and high) to drugs belonging to the NRTI (639 mutants) and NNRTI (747 mutants) drug classes. We considered a setting aimed at identifying amino acid mutations conferring high susceptibility (with respect to medium or low), and considered a mutant as highly susceptible to a drug class if it was annotated as being highly susceptible to at least one drug from that class.

Learning in first order logic

Our aim is to learn a first-order logic hypothesis for a target concept, i.e. mutation conferring resistance to a certain drug, and use it to infer novel mutations consistent with such hypothesis. We rely on definite clauses which are the basis of the Prolog programming language. A definite clause is an expression of the form `h` $\leftarrow$ `b1 AND ... AND bn`, where `h` and the `bi` are atomic literals. Atomic literals are expressions of the form `p(t1, ..., tn)` where `p/n` is a predicate symbol of arity `n` and the `ti` are terms, either constants (denoted by lower case) or variables (denoted by upper case) in our experiments. The atomic literal `h` is also called the head of the clause, typically the target predicate, and `b1 AND ... AND bn` its body. Intuitively, a clause represents that the head `h` will hold whenever the body `b1 AND ... AND bn` holds. For instance, a simple hypothesis like `res_against(A,nnrti)` $\leftarrow$ `mutation(A,C) AND close_to_site(C)` would indicate that a mutation `C` in the proximity of a binding site confers to mutant `A` resistance against a `nnrti`. Learning in this setting consists of searching for a set of definite clauses $H = \{c_i, ..., c_m\}$ covering all or most positive examples, and none or few negative ones if available. First-order clauses can thus be interpreted as relational features that characterize the target concept. The main advantage of these logic-based approaches with respect to other machine learning techniques is the expressivity and interpretability of the learned models. Models can be readily interpreted by human experts and provide direct explanations for the predictions.

Background knowledge

We built a relational knowledge base for the problem domain. Table 1 summarizes the predicates we included as a background knowledge. We represented the amino acids of the wild type with their positions in the primary sequence (`aa/2`) and the specific mutations characterizing them (`mut/4`). Target predicates were encoded as resistance of the mutation or mutant to a certain drug (`res_against/2`). Note that this encoding considers mutations at the amino acid rather than nucleotide level, i.e. a single amino acid mutation can involve up to three nucleotide changes. While focusing on single nucleotide changes would drastically expand the space of possible mutations, including the cost (in terms of nucleotide changes) of a certain amino acid mutation could help refining our search procedure.

Additional background knowledge was included in order to highlight characteristics of residues and mutations:

`typeaa/2` indicates the type of the natural amino

[1] http://www.hiv.lanl.gov/content/sequence/RESDB/
[2] downloadable at http://hivdb.stanford.edu/cgi-bin/GenoPhenoDS.cgi

acids according to the Venn diagram grouping based on the amino acids properties proposed in [14]. For example, a serine is a tiny and polar amino acid.

`color/2` indicates the type of the natural amino acids according to the coloring proposed in [15]. For example the magenta class includes basic amino acids as lysine and arginine while the blue class includes acidic amino acids as aspartic and glutamic acids. These groups of amino acids do not overlap as in the previous case.

`same_type_aa/3` indicates whether two residues belong to the same type `T`, i.e. a change from one residue to the other conserves the type of the amino acid.

`same_color_type/2` indicates whether two residues belong to the same coloring group, i.e. a change from one residue to the other conserves the coloring group of the amino acid.

`same_type_mut_t/3` indicates that a residue substitution at a certain position does not modify the amino acid type `T` with respect to the wild type. For example mutation i123v conserves the `aliphatic` amino acid type while mutation i123d does not (i.e. `different_type_mut_t/3` holds for it).

`same_color_type_mut/2` indicates that a residue substitution at a certain position does not modify the amino acid coloring group with respect to the wild type. For example mutation d123e conserves the `blue` amino acid group while mutation d123a does not (i.e. `different_color_type_mut/2` holds for it).

Other background knowledge facts and rules were added in order to express structural relations along the primary sequence and catalytic propensity of the involved residues:

`close_to_site/1` indicates whether a specific position is distant less than 5 positions from a residue belonging to a binding or active site. In our specific case, the background theory incorporates knowledge about a metal binding site and a heterodimerization site.

`location/2` indicates in which fragment of the primary sequence the amino acid is located. Locations are numbered from 0 by dividing the sequence into fragments of 10 amino acid lenght.

`catalytic_propensity/2` indicates whether an amino acid has a high, medium or low catalytic propensity according to [16].

`mutated_residue_cp/5` indicates how, in a mutated position, the catalytic propensity has changed (e.g. from low to high).

Algorithm overview

The proposed approach is sketched in Figure 1.

Step 1: Learning phase

The first step is the learning phase. An ILP learner is fed with a logical representation of the data $\mathcal{D}$ and of the domain knowledge $\mathcal{B}$ to be incorporated, and it returns a first-order logical hypothesis H for the concept of mutation conferring resistance to a certain class of inhibitors.

In this context there are two suitable ways to learn the target concept, depending on the type of input data and their labeling:

a) the one-class classification setting, learning a model from positive instances only. This is the approach we employ for Dataset 1: positive examples are mutations for which experimental prove is available that shows resistance to a drug, but no safe claim can be made on non-annotated mutations.

b) the binary classification setting, learning to discriminate between positive and negative instances. This setting is appropriate for Dataset 2: positive examples are in our experiments mutants labelled as highly susceptible to the drug class, negative examples are those with medium or low susceptibility.

The hypothesis is derived using the Aleph (A Learning Engine for Proposing Hypotheses) ILP system[3], which allows to learn in both settings. In the one-class classification case, it incrementally builds a hypothesis covering all positive examples guided by a

[3]http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/aleph.html

Bayesian evaluation function, described in [17], scoring candidate solutions according to an estimate of the Bayes' posterior probability that allows to trade-off hypothesis size and generality. In the binary classification case, Aleph builds the hypothesis covering all positive examples and none or few negative ones, guided by an m estimate evaluation function [18].

Aleph adds clauses to the hypothesis based on their coverage of training examples. Given a learned model, the first clauses are those covering most training examples and thus usually the most representative of the underlying concept.

In Figure 2 we show a simple example of learned hypothesis covering a set of training mutants from Dataset 2. The learned hypothesis models the ability of a mutation to confer the mutant resistance to NNRTI and is composed of four first-order clauses, each one covering different sets of mutations of the wild type as highlighted in colors: yellow for the first clause, blue for the second, red for the third, and green for the fourth one. Some mutations are covered by more than one clause as shown by the color overlaps. Bold letters indicate residues involved in the RT metal binding site (D110, D185 and D186) or the heterodimerization site (W401 and W414).

Step 2: Generative phase

The second step of our approach is the generative phase, in which the learned hypothesis is employed to find novel mutations that can confer drug resistance to an RT mutant. A set of candidate mutations can be generated by using the Prolog inference engine starting from the rules in the learned model. The rules are actually constraints on the characteristics that a mutation of the wild type should have in order to confer resistance to a certain inhibitor, according to the learned hypothesis.

Algorithm 1 details the mutation generation procedure. We assume, for simplicity, to have a model H for a single drug class. The procedure works by querying the Prolog inference engine for all possible variable assignments that satisfy the hypothesis clauses, each representing a mutation by its position and the amino acid replacing the wildtype residue. The set of mutations generated by the model is ranked according to a scoring function S_M before being returned by the algorithm. We defined S_M as the number of clauses in H that a candidate mutation m satisfies.

Referring to the model in Figure 2, for instance, the generated candidate mutations that satify the first two clauses are all the mutations changing the glycine in position 190 in a polar amino acid: 190Y, 190W, 190T, 190S, 190R, 190Q, 190N, 190K, 190H, 190E, 190D, 190C where for example the notation 190Y indicates a change of the wild type amino acid, located in position 190, into a tyrosine (Y). For instance, 190S and 190E in this list are already part of the known NNRTI surveillance mutations (see [19]).

Learning from mutations

We first learn general rules characterizing known resistance mutations (from Dataset 1) to be used for predicting novel candidate ones.

We divided the dataset of mutations into a training and a test set (70/30) in a stratified way, which means by preserving, both in the train and test set, the proportion of examples belonging to one of the two drug classes. The resulting training set is composed of a total of 106 mutations while the test set is composed of 45 mutations.

We trained the ILP learner on the training set and we evaluated on the test set the set of mutations generated using the learned model. The evaluation procedure takes the set of generated mutations and computes its enrichment in test mutations. We compare the recall of the approach, i.e. the fraction of test mutations generated by the model, with the recall of a baseline algorithm that chooses at random a set (of the same cardinality) of possible mutations among all legal ones.

Results averaged on 30 random splits of the dataset are reported in Table 2. On each split we performed 30 runs of our algorithm (Aleph has a random component generating the seed for the hypothesis search) and of the random generation algorithm in each one of the different learning tasks (NNRTI, NRTI). In this setting, the average sizes of the learned hypotheses for NNRTI and NRTI are 8 and 7 rules respectively. For each task we also report in column 3 and 4 of Table 2 the mean number of generated mutations over the 30 splits and the number of test set mutations for reference. In both cases, improvements are statistically significant according to a paired Wilcoxon test (α=0.01). Figure 3 gives further details about these results by showing how the mean recall on the test set varies if we compute it on the generated mutations satisfying only one clause, two clauses, three clauses and so on (x axis). The mean recall trend is shown in orange for the proposed generative approach and in green for a random generator of mutations for both classes of inhibitors.

We finally learned a model on the whole dataset in order to generate a single set of mutations for further inspection. We report five examples of novel mutations with the highest score for each one of the tasks: 90I, 98I, 103I, 106P, 179I for NNRTI and 60A, 153M, 212L, 229F, 239I for NRTI.

In [20], the authors found a set of novel mutations conferring resistance to efavirenz and nevirapine, which are NNRTI. Our mutation generation algorithm partially confirmed their findings. Mutation 90I was scored high (getting the maximum score of 5 satisfied clauses), mutation 101H was generated with a score of 3 out of 5, mutations 196R and 138Q with score 1 out of 5, while mutation 28K was not generated at all by our system as a candidate for conferring resistance to NNRTI.

An example of learned hypothesis is reported in Figure 4. For instance, according to the model, among the features a mutation should have for conferring resistance to a NNRTI, there is the change of a basic (magenta) residue of the wild type, e.g. lysine or arginine, into a residue with completely different phisico-chemical characteristics (rule 16).

Another example for the resistance to NNRTI is that a non conserved mutation is present in positions between 98 and 106 of the wild type sequence (rule 8).

Learning from mutants

The next set of experiments is focused on learning mutations from mutant data (Dataset 2). Learned models are still limited to single amino acid mutations, and so are novel mutants generated by the system.

We randomly assigned the mutants in Dataset 2 to 30 training and a test set splits, by avoiding having mutants containing the same resistance mutation (according to the labelling used in Dataset 1) in both training and test sets. For each of the 30 splits, we evaluated the recall of the generated mutations on the known resistance mutations (from Dataset 1), by first removing all the mutations that were also present in the training set. Comparison is again made on a baseline algorithm generating random legal mutations.

Results averaged on the 30 random splits are reported in Table 3. In this setting, the average sizes of the learned hypotheses for NNRTI and NRTI are 5 and 3 rules respectively. The small size of the models allows to substantially reduce the space of possible mutations to be generated. Our approach shows an average recall of 17% and 7% on the test mutations, which is statistically significantly higher than the recall obtained by randomly generating mutations (paired Wilcoxon test, α=0.01). Figure 5 gives further details about these results by showing the mean recall trend by varying the number of satisfied clauses of the generated mutations.

Figure 6 shows the hypothesis learned by the system when trained on the whole set of mutants in Dataset 2. It is easy to see that the hypothesis for the resistance to NNRTI identifies many (10 out of 19) of the known resistance survaillance mutations reported in [19]: 103N, 106A, 181C, 181I, 181V, 188C, 188H, 190A, 190E and 190S. In particular, mutation 181C has the highest predicted score as it satisfies four clauses of the learned model. Interestingly, other not previously reported mutations are suggested with a high score (3 out of 4 satisifed clauses) by the generative algorithm. These mutations may be novel mutations conferring resistance to NNRTI. For example, 181N, 181D, 318C and 232C.

29% of the mutations labeled as conferring resistance to NNRTI in Dataset 1 are also identified. Apart from the survaillance mutations those include also: 98G, 227C, 190C, 190Q, 190T and 190V. Key positions for the resistance to NNRTI, like 181 and 190 can be easily identified from the model in Figure 6, thanks to rules 15 and 23 that anchor the mutation to the specific position without restricting the change to a specific group of amino acids. A quite uncommon NNRTI-selected mutation, 238N, appears among the generated ones. Indeed, as reported in the summaries of the Stanford HIV Resistance Database, this mutations in combination with 103N, causes high-level resistance to nevirapine and efavirenz.

Also in the case of NRTI the generative algorithm suggests a few known survaillance mutations reported in [19]: 67E, 67G, 67N, 116Y, 184V, 184I (6 out of 34). 18% of the mutations labeled as conferring resistance to NRTI in Dataset 1 are also identified. Apart from known survaillance mutation those include also: 44D, 62V, 67A, 67S, 69R, 184T. Key positions for the resistance to NRTI as predicted by our model in Figure 6 are: 33, 67, 184, 194, 218. Of them, the effects of mutations in position 67 and especially in 184 have been well studied, while positions 33, 194, 218 seem to be novel. We found also that the predicted mutation 219H, which does not appear neither among the know survaillance mutations or among the resistance mutations in Dataset 1, is indeed reported in the Stanford HIV Database as a quite uncommon NRTI-selected mutation. Note that this uncommon mutation requires two nucleotide changes to emerge.

Discussion and Future Work

The results shown in the previous section are a promising starting point to generalize our approach to more complex settings. We showed that the approach scales from few hundreds of mutations as learning examples to almost a thousand of complete mutants. Moreover the learned hypotheses significantly constraint the space of all possible single amino acid mutations to be considered, paving the way to the expansion of the method to multi-site mutant generation. This represents a clear advantage over alternative existing machine learning approaches, which would require the preliminary generation of all possible mutants for their evaluation. Restricting to RT mutants with two mutated amino acids, this would imply testing more than a hundred million candidate mutants. At the same time this simple ILP approach cannot attain the same accuracy levels of a pure statistical approach. We are currently investigating more sophisticated statistical relational learning approaches for improving the accuracy of the learned models and for assigning weights to the clauses to be used for selecting the most relevant generated mutations. These can be used as a pre-filtering stage, producing candidate mutants to be further analysed by complex statistical approaches, and additional tools evaluating, for instance, their stability. An additional direction to refine our predictions consists of jointly learning models of resistance to different drugs (e.g. NNRTI and NRTI), possibly further refining the joint models on a per-class basis. On a predictive (rather than generative) task, this was shown [21] to provide improvements over learning distinct per-drug models.

The approach is not restricted to learning drug-resistance mutations in viruses. More generally, it can be applied to learn mutants having certain properties of interest, e.g. improved or more specific activity of an enzyme with respect to a substrate, in a full protein engineering fashion.

Conclusions

In this work we proposed a simple relational learning approach applicable to evolution prediction and protein engineering. The algorithm relies on a training set of mutation data annotated with drug resistance information, builds a relational model characterizing resistant mutations, and uses it to generate novel potentially resistant ones. Encouraging preliminary results on HIV RT data indicate a statistically significant enrichment in resistance conferring mutations among those generated by the system, on both mutation-based and mutant-based learning settings. Albeit preliminary, our results suggest that the proposed approach for learning mutations has a potential in guiding mutant engineering, as well as in predicting virus evolution in order to try and devise appropriate countermeasures. A more detailed background knowledge, possibly including 3D information whenever available, is necessary in order to further focus the set of generated mutations, and possibly post-processing stages involving mutant evaluation by statistical machine learning approaches [5]. In the next future we also plan to generalize the proposed approach to jointly generate sets of related mutations shifting the focus from the generation of single amino acid mutations to mutants with multiple mutations.

Acknowledgements

EC and TL acknowledge the support of the F.W.O. (Belgium) and the F.R.S.-F.N.R.S. (Belgium). ST and AP acknowledge the support of the Italian Ministry of University and Research under grant PRIN 2009LNP494 (Statistical Relational Learning: Algorithms and Applications).

References

1. Cao ZW, Han LY, Zheng CJ, Ji ZL, Chen X, Lin HH, Chen YZ: **Computer prediction of drug resistance mutations in proteins REVIEWS**. *Drug Discovery Today: BIOSILICO* 2005, **10**(7).

2. Rubingh DN: **Protein engineering from a bioindustrial point of view**. *Current Opinion in Biotechnology* 1997, **8**(4):417–422.

3. Muggleton S, De Raedt L: **Inductive Logic Programming: Theory and Methods**. *University Computing* 1994, :629–682.

4. Cilia E, Passerini A: **Frankenstein Junior: a relational learning approach toward protein engineering**. In *Proceedings of the Annotation, Interpretation and Management of Mutations (AIMM) Workshop@ECCB2010* 2010.

5. Capriotti E, Fariselli P, Rossi I, Casadio R: **A three-state prediction of single point mutations on protein stability changes.** *BMC bioinformatics* 2008, **9 Suppl 2**:S6.

6. Needham CJ, Bradford JR, Bulpitt AJ, Care Ma, Westhead DR: **Predicting the effect of missense mutations on protein function: analysis with Bayesian networks.** *BMC bioinformatics* 2006, **7**:405.

7. Bromberg Y, Rost B: **SNAP: predict effect of non-synonymous polymorphisms on function.** *Nucleic acids research* 2007, **35**(11):3823–35.

8. Rhee SY, Taylor J, Wadhera G, Ben-Hur A, Brutlag DL, Shafer RW: **Genotypic predictors of human immunodeficiency virus type 1 drug resistance.** *Proceedings of the National Academy of Sciences of the United States of America* 2006, **103**(46):17355–60.

9. Götte M, Li X, Wainberg M: **HIV-1 reverse transcription: a brief overview focused on structure-function relationships among molecules involved in initiation of the reaction**. *Archives of biochemistry and biophysics* 1999, **365**(2):199–210.

10. Shafer R: **Rationale and Uses of a Public HIV Drug-Resistance Database**. *Journal of Infectious Diseases* 2006, **194**(Supplement 1):S51–S58.

11. De Clercq E: **HIV inhibitors targeted at the reverse transcriptase**. *AIDS research and human retroviruses* 1992, **8**(2):119–134.

12. Spence R, Kati W, Anderson K, Johnson K: **Mechanism of inhibition of HIV-1 reverse transcriptase by nonnucleoside inhibitors**. *Science* 1995, **267**(5200):988–993.

13. Richter L, Augustin R, Kramer S: **Finding Relational Associations in HIV Resistance Mutation Data**. In *Proceedings of Inductive Logic Programming (ILP), Volume 9* 2009.

14. Betts M, Russell R: **Amino-Acid Properties and Consequences of Substitutions**. *Bioinformatics for geneticists* 2003, :311–342.

15. Taylor WR: **The classification of amino acid conservation.** *Journal of Theoretical Biology* 1986, **119**(2):205–218.

16. Bartlett G, Porter C, Borkakoti N, Thornton J: **Analysis of catalytic residues in enzyme active sites**. *Journal of Molecular Biology* 2002, **324**:105–121.

17. Muggleton S: **Learning from Positive Data**. In *Inductive Logic Programming Workshop* 1996:358–376.

18. Džeroski S, Bratko I: **Handling noise in Inductive Logic Programming**. In *ILP92*, Report ICOT TM-1182. Edited by Muggleton S 1992.

19. Bennett DE, Camacho RJ, Otelea D, Kuritzkes DR, Fleury H, Kiuchi M, Heneine W, Kantor R, Jordan MR, Schapiro JM, Vandamme AM, Sandstrom P, Boucher CaB, van de Vijver D, Rhee SY, Liu TF, Pillay D, Shafer RW: **Drug resistance mutations for surveillance of transmitted HIV-1 drug-resistance: 2009 update.** *PloS one* 2009, **4**(3):e4724.

20. Deforche K, Camacho RJ, Grossman Z, Soares Ma, Van Laethem K, Katzenstein Da, Harrigan PR, Kantor R, Shafer R, Vandamme AM: **Bayesian network analyses of resistance pathways against efavirenz and nevirapine.** *AIDS (London, England)* 2008, **22**(16):2107–15.

21. Cilia E, Landwehr N, Passerini A: **Relational Feature Mining with Hierarchical Multitask kFOIL**. *Fundamenta Informaticae* 2011, **113**(2):151–177.

22. Theys K, Deforche K, Beheydt G, Moreau Y, van Laethem K, Lemey P, Camacho RJ, Rhee SY, Shafer RW, Van Wijngaerden E, Vandamme AM: **Estimating the individualized HIV-1 genetic barrier to resistance using a nelfinavir fitness landscape.** *BMC bioinformatics* 2010, **11**:409.

Algorithms

Algorithm 1 - Mutation generation algorithm.

Algorithm for novel relevant mutations discovery.

Algorithm 1 Mutation generation algorithm.

```
1:  input: background knowledge $\mathcal{B}$, learned model $H$
2:  output: rank of the most relevant mutations $\mathcal{R}$
3:  procedure GenerateMutations($\mathcal{B}, H$)
4:      Initialize $\mathcal{D}_{\mathcal{M}} \leftarrow \emptyset$
5:      $A \leftarrow$ find all assignments $a$ that satisfy at least one clause $c_i \in H$
6:      for $a \in A$ do
7:          $m \leftarrow$ mutation corresponding to the assignments $a \in A$
8:          $score \leftarrow S_M(m)$                      ▷ number of clauses $c_i$ satisfied by $a$
9:          $\mathcal{D}_{\mathcal{M}} \leftarrow \mathcal{D}_{\mathcal{M}} \cup \{(m, score)\}$
10:     end for
11:     $\mathcal{R} \leftarrow$ RankMutations($\mathcal{D}_{\mathcal{M}}, \mathcal{B}, H$)          ▷ rank relevant mutations
12:     return $\mathcal{R}$
13: end procedure
```

Figures

Figure 1 - Mutation engineering algorithm.

Schema of the mutation engineering algorithm.

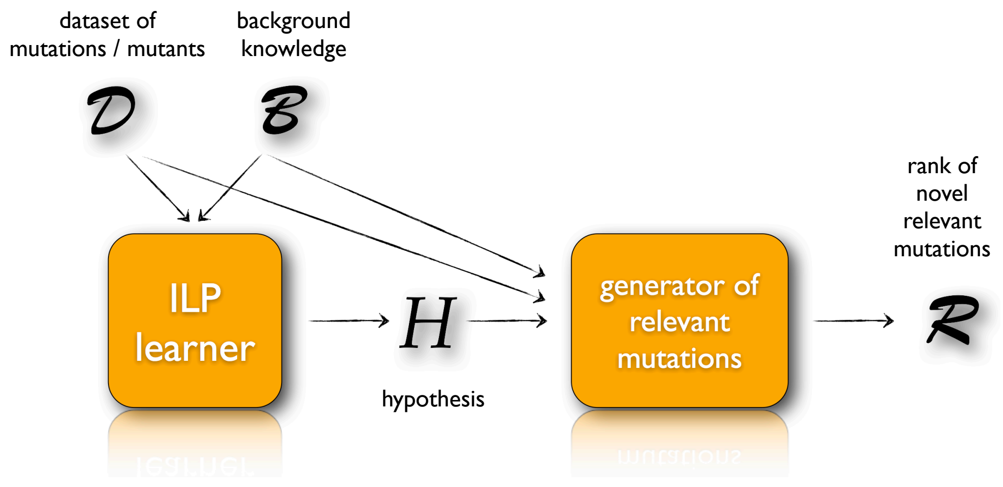
dataset of mutations / mutants
background knowledge
D
B
ILP learner
H
hypothesis
generator of relevant mutations
rank of novel relevant mutations
R

Figure 2 - Model for the resistance to NNRTI learned from Dataset 2
An example of learned hypothesis for the NNRTI task with highlighted examples covered by the hypothesis clauses.

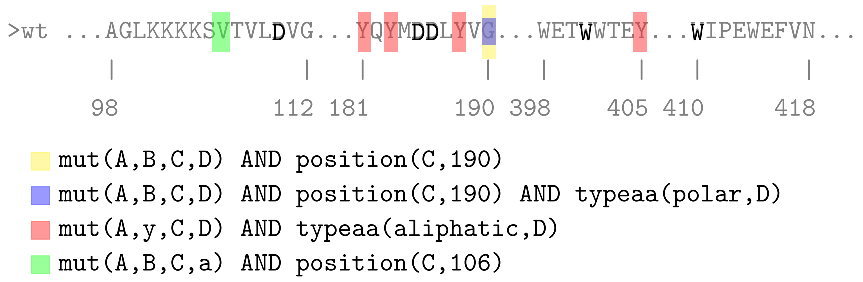
>wt ...AGLKKKKSVTVLDVG...YQYMDDLYVG...WETWWTEY...WIPEWEFVN...
98
112
181
190
398
405
410
418
mut(A,B,C,D) AND position(C,190)
mut(A,B,C,D) AND position(C,190) AND typeaa(polar,D)
mut(A,y,C,D) AND typeaa(aliphatic,D)
mut(A,B,C,a) AND position(C,106)

Figure 3 - Mean recall trend by number of satisfied clauses (Dataset 1)
Mean recall of the generated mutations on the resistance test set mutations from Dataset 1 by varying the number of satisfied clauses. The mean recall values in orange refer to the proposed generative algorithm. The mean recal values in green refer to a random generator of mutations.

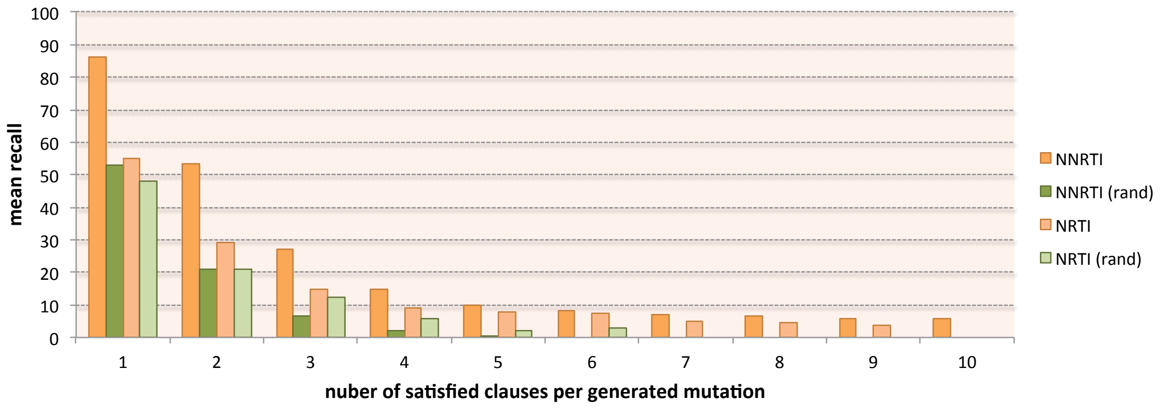

Figure 4 - Example of model learned from Dataset 1
Model learned for NRTI and NNRTI on the dataset mutations Dataset 1. Note that in this case A is a variable that identifies a single amino acid mutation.

```
1-res_against(A,nrti) ← mut(A,B,C,D) AND color(red,D)
2-res_against(A,nrti) ← mut(A,B,C,D) AND color(red,D) AND color(red,B)
3-res_against(A,nrti) ← mut(A,B,C,D) AND location(7.0,C) AND mutated_residue_cp(C,medium,medium)
4-res_against(A,nrti) ← mut(A,B,C,D) AND location(7.0,C)
5-res_against(A,nrti) ← same_color_type_mut(A,B)
6-res_against(A,nrti) ← mut(A,B,C,D) AND mutated_residue_cp(C,medium,high)
7-res_against(A,nrti) ← mut(A,B,C,D) AND mutated_residue_cp(C,medium,small)
8-res_against(A,nnrti) ← different_color_type_mut(A,B) AND location(11.0,B)
9-res_against(A,nnrti) ← mut(A,B,C,D) AND mutated_residue_cp(C,small,small)
10-res_against(A,nnrti) ← same_color_type_mut(A,B)
11-res_against(A,nnrti) ← mut(A,v,C,D)
12-res_against(A,nnrti) ← mut(A,B,C,D) AND location(15.0,C)
13-res_against(A,nnrti) ← mut(A,B,C,i)
14-res_against(A,nnrti) ← mut(A,B,C,D) AND location(11.0,C)
15-res_against(A,nnrti) ← mut(A,B,C,D) AND color(red,D)
16-res_against(A,nnrti) ← mut(A,B,C,D) AND color(magenta,B) AND different_color_type_mut(A,C)
17-res_against(A,nnrti) ← mut(A,B,C,D) AND location(21.0,C)
```

Figure 5 - Mean recall trend by number of satisfied clauses (Dataset 2)
Mean recall of the generated mutations on the resistance test set mutations from Dataset 2 by varying the number of satisfied clauses. The mean recall values in orange refer to the proposed generative algorithm. The mean recal values in green refer to a random generator of mutations.

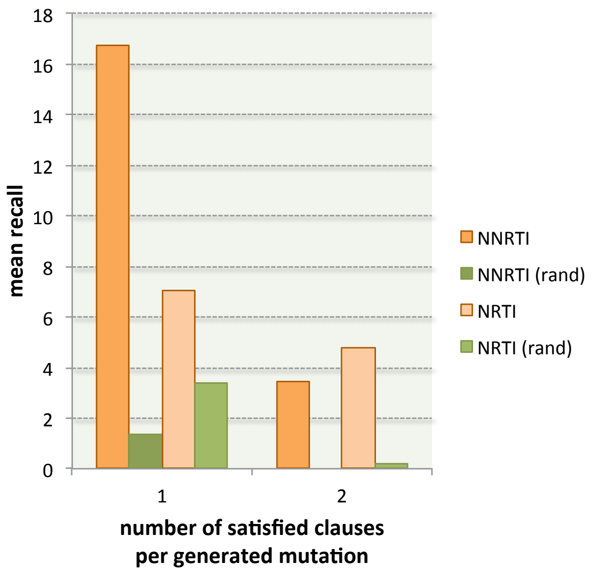

Figure 6 - Model learned from the whole Dataset 2

Model learned for NRTI and NNRTI on the whole dataset of mutants Dataset 2. Note that in this case A is a variable that identifies a mutant.

```
1-res_against(A,nrti) ← mut(A,B,C,D) AND position(C,67)
2-res_against(A,nrti) ← mut(A,B,C,g) AND color(red,B)
3-res_against(A,nrti) ← mut(a,B,C,D) AND typeaa(nonpolar,D)
4-res_against(A,nrti) ← mut(A,B,C,D) AND position(C,33)
5-res_against(A,nrti) ← mut(A,B,C,D) AND position(C,218)
6-res_against(A,nrti) ← mut(A,B,C,h) AND catalytic_propensity(B,high)
7-res_against(A,nnrti) ← mut(A,B,C,D) AND position(C,194)
8-res_against(A,nnrti) ← mut(A,B,C,D) AND position(C,184)
9-res_against(A,nnrti) ← mut(A,B,C,D) AND catalytic_propensity(D,high) AND typeaa(neutral,D) AND
typeaa(nonpolar,B)
10-res_against(A,nrti) ← mut(A,B,C,d) AND position(C,44)
11-res_against(A,nrti) ← mut(A,B,C,r) AND typeaa(tiny,B) AND typeaa(polar,B)
12-res_against(A,nrti) ← mut(A,d,C,D) AND catalytic_propensity(D,medium)
13-res_against(A,nrti) ← mut(A,d,C,D) AND catalytic_propensity(D,medium) AND typeaa(polar,D)
14-res_against(A,nrti) ← mut(A,B,C,k) AND position(C,203)
```

```
15-res_against(A,nnrti) ← mut(A,B,C,D) AND position(C,190)
16-res_against(A,nnrti) ← mut(A,B,C,a) AND position(C,106)
17-res_against(A,nnrti) ← mut(A,B,C,D) AND same_typeaa(D,B,tiny) AND same_typeaa(D,B,nonpolar)
18-res_against(A,nnrti) ← mut(A,B,C,D) AND catalytic_propensity(D,high) AND typeaa(aromatic,B) AND
same_typeaa(D,B,neutral)
19-res_against(A,nnrti) ← mut(A,B,C,n) AND position(C,103)
20-res_against(A,nnrti) ← mut(A,B,C,D) AND catalytic_propensity(B,high) AND typeaa(neutral,B) AND
same_typeaa(D,B,polar)
21-res_against(A,nnrti) ← mut(A,B,C,D) AND position(C,177) AND catalytic_propensity(D,medium) AND
same_type_mut_t(A,C,polar)
22-res_against(A,nnrti) ← mut(A,B,C,n) AND position(C,238)
23-res_against(A,nnrti) ← mut(A,B,C,D) AND position(C,181)
24-res_against(A,nnrti) ← mut(A,y,C,D) AND typeaa(small,D)
```

Tables

Table 1 - Background knowledge predicates

Summary of the background knowledge facts and rules. MutID is a mutation or a mutant identifier depending on the type of the learning problem.

Background Knowledge Predicates	
`aa(Pos,AA)`	indicates a residue in the wild type sequence
`mut(MutID,AA,Pos,AA1)`	indicates a mutation: mutation or mutant identifier, position and amino acids involved, before and after the substitution
`res_against(MutID,Drug)`	indicates whether a mutation or mutant is resistant to a certain drug
`color(Color,AA)`	indicates the coloring group of a natural amino acid
`typeaa(T,AA)`	indicates the type (e.g. aliphafatic, charged, aromatic, polar) of a natural amino acid
`same_color_type(R1,R2)`	indicates whether two residues belong to the same coloring group
`same_typeaa(R1,R2,T)`	indicates whether two residues are of the same type `T`
`same_color_type_mut(MutID, Pos)`	indicates a mutation to a residue of the same coloring group
`different_color_type_mut(MutID, Pos)`	indicates a mutation changing the coloring group of the residue
`same_type_mut_t(MutID, Pos, T)`	indicates a mutation to a residue of the same type `T`
`different_type_mut_t(MutID, Pos)`	indicates a mutation changing the type of the residue
`close_to_site(Pos)`	indicates whether a specific position is close to a binding or active site if any
`location(L,Pos)`	indicates in which fragment of the primary sequence the amino acid is located
`catalytic_propensity(AA,CP)`	indicates whether an amino acid has a high, medium or low catalytic propensity
`mutated_residue_cp(Rw,Pos,Rm,CPold,CPnew)`	indicates how, in a mutated position, the catalytic propensity has changed (e.g. from low to high)

Table 2 - Results (Dataset 1)

Statistical comparisons of the performance of the proposed algorithm with an algorithm generating mutations at random. The average recall has been computed for each one of the learning tasks over the 30 splits by averaging recall over 30 repeated runs of the two algorithms. Results of a paired Wilcoxon test ($\alpha = 0.01$) on the statistical significance of the performance differences are also reported. A black bullet indicates a statistical significant improvement of our algorithm over a random generator. The mean number of generated mutations over the 30 splits and the number of test mutations is reported on the side.

	Mean recall % on 30 splits			
	Algorithm	**Random Generator**	*Mean n. generated mutations*	*n. test mutations*
NNRTI	86 •	58	5201	17
NRTI	55 •	46	5548	28

Table 3 - Results (Dataset 2)

Statistical comparisons of the performance of the proposed algorithm with an algorithm generating mutations at random. The average recall has been computed for each one of the learning tasks over the 30 splits. Results of a paired Wilcoxon test ($\alpha = 0.01$) on the statistical significance of the performance differences are also reported. A black bullet indicates a statistical significant improvement of our algorithm over a random generator. The mean number of generated mutations and the mean number of test mutations over the 30 splits is reported on the side.

	mean recall %			
	Algorithm	**Random Gen.**	*mean n. generated mutations*	*mean n. of test mutations*
NNRTI	17 •	1	236	26
NRTI	7 •	3	420	40