A Machine Learning Approach to Predicting HIV Progression

Do June Min DMIN1@SWARTHMORE.EDU

Yang Yi YYI1@SWARTHMORE.EDU

Abstract

HIV is a worldwide epidemic that has caused over 25 million deaths worldwide. Like many viruses, its effects vary from individual to individual and is largely determined by ones genetic underpinning. Given patient genomic data, we propose utilizing bioinformatic techniques such as MSA and Khmer to generate feature vectors which we can input into machine learning algorithms to predict a given patients outcome. This information will be useful in determining which patients have a good chance of recovery, which patients will probably never recover, and which borderline patients would benefit the most from heightened monitoring. Overall, our results show that performing a MSA and using minimum entropy to isolate feature vectors produces the most predictive model.

1. Introduction

HIV is not a one size fits all virus and previous work has established that various genetic strains of the virus are correlated with virus severity. For any given case of HIV, the nucleotide sequences that comprise that unique case contains a wealth of information including whether or not the patient can reasonably be expected to ultimately recover from the virus. In an ideal world, doctors would be able to accurately predict patient outcomes given only the viruses genetic sequences and a few other pieces of information such as a patients viral load and CD4 count at the beginning of therapy.

This problem can be framed as a mapping; giving a set of features as input, we want to output the most likely result. Framed this way, this problem seems well suited for machine learning, but in order to successfully utilize machine learning classifiers, several challenges must be overcome. First, we must find a way to convert the data into meaningful features; to do so, we decided to try two approaches: MSA and Khmer. Performing an MSA allowed us to be sure that the columns between sequences were meaningfully aligned which allowed us to use algorithms such as minimum entropy to find potentially significant column vectors that we could use as features. In addition, in the event that the data proved unconducive to MSA, we also implemented Khmer which does not require sequences to be aligned. From k = 3 to k = 9, we utilized Khmer to obtain a feature vector that obtained the counts of all unique substrands.

Finally, after successfully transforming the genomic data into a feature vector, we input it into machine learning classifiers, gauging each's accuracy. We implemented seven unique classifiers (KNN, neural networks, decision trees, SVMs, Naive Bayes, AdaBoost, and Random Forest). To better analyze each classifier's predictions, we computed several analytical metrics: accuracy, precision, recall, F score, and Mathew's correlation coefficient.

Overall, we found that combining MSA with minimum entropy produced the most effective feature vectors and that using khmer feature vectors did not enhance predictive accuracy. We also found that AdaBoost was the most predictive machine learning classifier followed closely by a linear SVM.

2. Related Work

2.1. Computational Biology: HIV Protease

Recognizing that a genetic understanding of HIV will greatly assist efforts in combating the disease, computational biologists have conducted many tests to test hypotheses of how HIV-1 affects human proteins. Notably, in 2015, Rognvaldsson et al conducted a machine learning experiment designed to classify of HIV-1 protease profiles within patients (within our data, this is the PR sequence). For background, HIV-1 protease is short for retroviral aspartyl protease (retropepsin) which is essential for the life-cycle of HIV. Without effective HIV protease, HIV virions remain infectious and mutation of HIV protease's active site or inhibition of its activity disrupts HIVs ability to repli-

CPSC 68 Bioinformatics Final Report Proceedings, Swarthmore College, Fall 2014.

cate and infect additional cells. Thus, while not completely analogous to predicting patient outcomes, their work is extremely correlated to ours as a patients HIV-1 protease profile provides a strong indication of whether or not they will eventually recover.

Rognvaldsson et al discovered that a linear SVM with standard orthogonal encoding is the best predictor across all data sets. In our experiments, we hope to build upon their work. Within their experiments they focused heavily on training SVMs and compared across two online predictors, HIVcleave and PROSPER. Instead, we aim to compare SVMs against a wider variety of machine learning classifiers including but not limited to neural networks, decision trees, and Naive Bayes. We also aim to divide the data set into alternative features from that of their paper by utilizing different processing techniques such as MSA and Khmer.

2.2. Biological Indicators

In addition to building a robust model, we hope to produce helpful references that can explain whether our models predictive power is biologically sound. Biologists Langford et al in 2007 published a much more analytical paper on HIV progression in patients, placing less emphasis on algorithms and more on biologically observable metrics such as CD4 cell count, HIV-RNA, and host genetics. Later in 2015, Poorolajal et al released a study which examined 2,473 HIV-infected patients and examined disease progression throughout a 1, 5, and 10-year period. Their findings complemented that of the Langford study and showed that in addition to non-modifiable predictors such as age and sex, there was a significant association with decreased levels of CD4 count (P = 0.001). Their work presents an alternative approach to that of that of Rognvaldsson's- that HIV progression can also be predicted by isolating causal factors and performing regression.

2.3. Hybrid Approaches

In 2009, Korenromp et al measured fluctuations on chronically infected patients and estimated that 55 percent of population-level variation in RNA, and 75 percent of variation in CD4 were significant in quantifying risk levels for any given patient. The results show that sequence analysis can definitely yield predictive features; however, because these mutations behave different under various conditions, it is challenging to find a consistent pattern within the noise. Here, computation approaches can play a pivotal role. Carvajal-Rodrguez believes that computational tools are the cheapest and most efficient way to analyze individual genomic data which can then be used to tailor treatments to individual patients. In 2007, his study demonstrated that trained properly, computational tools can not only successfully identify relevant mutations, but also through a prediction system, recommend drugs that the mutations are most susceptible to.

At times, biology and bioinformatics can appear disjoint as biologists generally preferred observable phenomenon over black box machine learning models. However, both viewpoints contain merit and eschewing one compromises the potential value of results. Within our project, we hope to do the same by providing the biological intuition behind our algorithmic work whenever possible, such as by analyzing whether our decision tree splits occur at sites that are traditionally associated with HIV mutations. In doing so, we aspire to produce results that members of both fields would be willing to adopt.

3. Methodology

Our program contains various methods, each of which falls into one of three categories: data processing, classification, and analysis.

3.1. Data Processing: Initial Features

For training, we received a data set consisting of approximately 1000 cases and for testing, a data set containing approximately 750 more. Each entry contained four pieces of patient data: their reverse transcriptase (RT) sequence, their, protease (PR) sequence, their viral load, and their CD4 count at the beginning of therapy. Some of the data entries were incomplete; in this case, we disregarded them completely.

Viral load and CD4 count were already discretized variables and thus required no processing. We simply added them as features in their current state. But as previously mentioned, the RT and PR sequences required refinement as they were both far too long to feasibly use.

3.2. Data Processing: MSA

Nucleotide sequences are hundreds of characters long and many sections of the sequences are homogenous amongst patients who recover and those who dont. However, we cannot just start by eliminating columns from the training data as the presence of gaps disjoints the inter-patient sequence alignments and introduces a litany of offset errors. Somewhere along the sequence, gaps may cause every base in one sequence to differ from those in another when in reality, the ordering and bases are identical sans gaps.

This forms the crux of why we performed an MSA, which we ran on both training and testing cases; we wanted the bases within a given column to be significant relative to other bases in similar columns. We could then perform minimum entropy to identify columns where the patients cases showed the most genetic disparities and be more con-

fident that these disparities were biologically significant, not products of random offsets.

To perform the MSA, we utilized ClustalW, an open library tool that takes in a file of genomic sequences and returns an updated file of sequences with the columns properly aligned. After obtaining the result, we ran our own minimum entropy implementation on each of the result columns and ranked them based on genetic disparity. For reference, our minimum entropy algorithm can be found in Appendix Figure A. We then took only the top forty ranked columns and appended them to our feature vector; this greatly reduced the amount of features we need to consider relative to using the entire genetic sequence and allowed us to disregard homogenous columns that were unlikely to enhance accuracy. Doing so also made our feature vector much less susceptible to the curse of dimensionality, which was significant given that we had less than one thousand training cases.

3.3. Data Processing: K-mer

However, recognizing that MSA had its shortcomings, such as the fact that whenever we obtain a new case, we need to re-align the sequences (and hence may change the sites with the most entropy), we decided to also adopt an approach that did not require any alignment, k-mer. k-mer works by taking in a length, k, as input, and then returning a count of all unique sub-strands of length k within a greater strand. For example, if the strand was ATCG and k = 1, then k-mer would return [A: 1, T: 1, C:1, G:1]. We then transformed this output into one large feature vector by creating a dictionary which maps each index within the array to a length k substrand, and where the value contained in that index represents the number of times the substrand appears. Then, a machine learning algorithm could naturally parse out which indices contained values that were significant, which we could then lookup in our dictionary to identify the associated nucleotide sequences. Finally, since it is mainly returning a count within each individual sequence in vacuum, we do not have to deal with any issues related to alignment.

To implement k-mer, we installed Khmer, an online k-mer script that takes in a sequence and value k as input and outputs all subsequences of length k as well as their frequency count. After running the script, we obtained the top 20 most frequent 9-mers for each sequence and converted them into features by letting the subsequence represent the dimension and the count represent the value. We then inserted these features into our aggregate feature vector which contains all the other features we obtained from the data and MSA.

3.4. Data Processing: Data Augmentation

Finally, we realized that the training set was not representative of the test set as within the training set there are many more negative examples than positive ones. The test set, on the other hand, appears much more evenly split. Concerned that our classifiers might attempt to take advantage of this imbalance and opt to gain accuracy by uniformly outputting negative examples, we experimented with weighting positive examples with integer weights $w = 2$ and $w = 3$; that is, while processing the data, we added each positive example to the training data w times as opposed to once. We then performed all of the aforementioned data processing, only this time with the newly augmented data set.

3.5. Data Processing: Parameters

In using the data processing and augmentation methods we have mentioned above, we have chosen the parameters shown in the following table.

To deal with the imbalance between negative and positive examples, we sampled each positive data in the training set $w = 3$ times.

Also, we use minimum entropy to measure the degree of variation within a column. 40 columns with highest variation are considered.

Finally, we settled on doing 7-mer counting, with the 20 most frequent 7-words considered as features.

Table 1. Data augmentation Parameters

Data Processing Method	Parameters
Weighting	w=3
MSA	Number of Columns = 40, Measure: Minimum Entropy
k-mer counting	k=7, Number of k-mers: 20

As a result, each data point is transformed from a 4-vector to a 122-vector.

3.6. Classification

In total, we utilized 7 classifiers: KNN, neural networks, decision trees, SVMs, Naive Bayes, AdaBoost, and Random Forest. For brevity, in this section we will only explain the two classifiers that proved most instrumental to our results: SVM and AdaBoost. An additional table explaining the rest within the context of our study can be found in Appendix Figure B.

3.7. Classification: SVMs

3.8. Classification: AdaBoost

AdaBoost, short for adaptive boosting, is one of the many machine learning algorithms we imported from Sklearn. The issue with weak classifiers is that they can often only learn rules of thumbs that make them more effective predictors than random guess, but are not nuanced enough to boost them into the upper echelons of predictive accuracy. AdaBoost strives to boost their capabilities by weighting training examples, prioritizing harder to train cases over easier ones. This process can be expressed as the following algorithm.

Given training data $(x_1, y_1), .., (x_m, y_m)$
$y_1 \in \{-1, +1\}$ $x_i \in X$ is the object or instance, y_i is the label.
For $t = 1, .., T$
create distribution D_t on $\{1, ..., m\}$
select weak classifier with smallest error ϵ_t on D_t.
$\epsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i]$
$h_t : X \rightarrow \{-1, +1\}$
output single classifier $H_final(x)$.

This weighting is particularly useful for our task because the crux of our model's predictive capability will lie in its ability to classify patients with unique or borderline features. Some patients will be extremely easy to classify and can probably be classified with a very high degree of accuracy based on one feature alone (i.e. patients with significantly higher viral loads are almost guaranteed to experience HIV progression). It is the outlier cases in which we must incorporate multiple more dimensions such as those pertaining to genetic sequences, cases that plague most of our weak classifiers and where AdaBoost can provide the most impact. By weighting these cases, we can find more specific rules that don't necessaryily apply to the entire data set, but are particularly predictive in borderline cases. Because we already have two dimensions, CD4 cell count and viral load, that are excellent predictors of easy-to-classify cases, it is reasonable to assume that by performing such weighting, our classifier will not unlearn many previous cases.

3.9. Classification: Parameters

In addition to SVMs and AdaBoost, the parameters for the rest of classifiers can be found below. As previously mentioned, if interested in our reasoning for using each algorithm, see Appendix Figure B.

Table 2. Algorithm Parameters

Algorithms	Parameters
Gauss NB	Default Parameters
Multi NB	Default Parameters
SVM Linear	C = 10.0
SVM RBF	C = 10.0
SVM Sigmoid	C = 10.0
Decision Tree	Default Parameters
Neural Network	Architecture and training parameters included in the appendix
AdaBoost	100 base learners
Random Forest	100 base learners

TODO: explanation

3.10. Analysis: Confusion Matrix

Although we would like to output a soft vector that probabilistically corresponds to various outcomes of HIV progression, due to the nature of our data set, our outputs are essentially binary (we only know whether the virus either regressed or progressed). Therefore, the majority of our analysis focuses on binary statistics such as precision and recall. Calculating precision and recall required obtaining rates of true and false positives/negatives, which are contained in a confusion matrix in the manner shown below.

Table 3. The composition of a confusion matrix

Truth Prediction	Negative	Positive
Negative	True Negative	False Positive
Positive	False Negative	True Positive

For brevity, we refer to them as TP, TN, FP, FN.

Although confusion matrices do not explicitly compute precision and recall, one can easily obtain them using the following equation.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

3.11. Analysis: F-Score

We then utilized precision and recall to obtain an F-score, which provides us with a more holistic alternative to accuray, considering both the precision and the recall (sensitivity). Precision is computed by dividing the number of correct positive results by the number of all positive results and recall is computed by dividing the number of correct positive results by the number of all possible correct positive

results. The F-Measure is then computed via a weighted average of the two and takes on a value between 0 and 1.

Specifically, F-score is calculated as follows:

$$\text{F-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} = \text{recall}}$$

The precision component of the F-Score makes it an interesting supplement to accuracy. Accuracy measures how close one can get to the true value while precision measures the closeness between measurements (an analogy can be found in shooting a basketball where accuracy is getting the ball close to the basket while precision is getting the ball to go to the same spot every time regardless of whether that spot is close to the basket). Precision is important because it measures how reproducible the results are, and with respect to tools such as diagnostic tests, we want them to contain lower volatility. Having precise tests make it easy to detect outliers, increasing our confidence that our results are biologically significant.

3.12. Analysis: MCC

Finally, we augmented the F-score by calculating Mathews Correlation Coefficient which differs from F-score in that it accounts for true negatives, which makes MCC more robust to imbalances in data. MCC takes on values between -1 and +1 where +1 represents perfect prediction and -1 represents total inaccuracy. We calculated it as follows:

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}$$

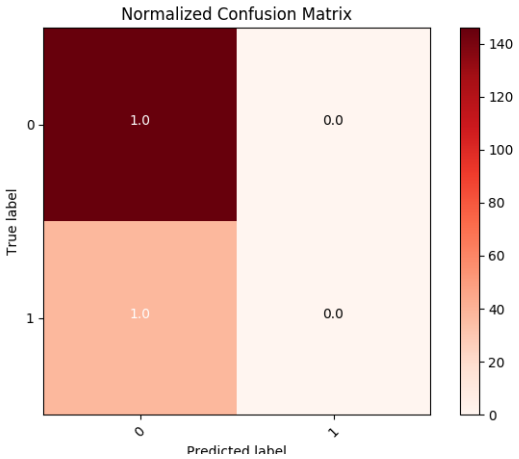
4. Results

Below are the results for each classifier.

4.1. Results: Confusion Matrix

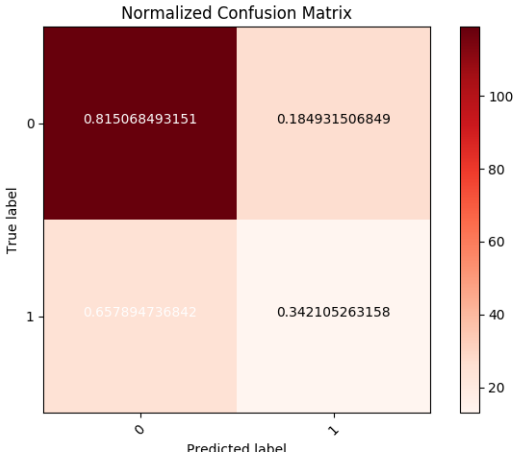
Analyzing the confusing matrix yielded helpful insights as to how our classifiers were getting predictions right. For instance, the decision tree for neural networks confirmed our intuition that due to the heavy imbalance of progression cases in our training set, certain classifiers attempted to gain accuracy by classifying all cases under one label (Figure 1).

Figure 1. Normalized Confusion Matrix of Neutral Network Classification Result



We could also see how well our data processing improved our results. After performing data augmentation and weighting regression cases to balance our training set, the same neural network algorithm produced the following confusion matrix (Figure 2).

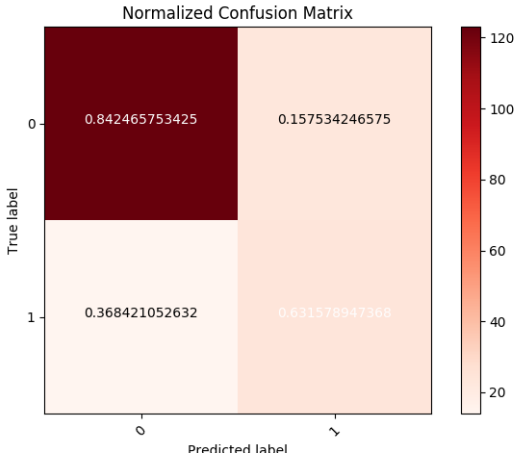
Figure 2. Normalized Confusion Matrix of Decision Tree Classification Result



While the overall accuracy was slightly less in this case, we feel much more confident in the latter neural network's ability to generalize to novel data. This is a helpful example as to why accuracy is not the end all be all. We then analyzed the matrix of our best performer, AdaBoost (Figure 3). Based on the matrix, we conclude the best classifiers such as AdaBoost were able to discern some patterns within the data noise. The composition of false negatives and false positives was also much more balanced than that of Figure 2 which suggests that it is not overcompensating to accommodate the training set. Despite this, we still

feel that some of the training set skew is still present as the prevalence of false negatives relative to false positives is higher than we'd like it to be.

Figure 3. Normalized Confusion Matrix of AdaBoost Classification Result



4.2. Results: Evaluation Metrics

In addition to confusion matrices, evaluative metrics such as accuracy, precision, recall, F-Score, and Mathews Correlation Coefficient can also provide helpful insights. The metrics for each classifier can be found in Table 4. For each metrics, the cell(classifier) with highest value is emboldened.

Table 4. Evaluation table of classification results using different evaluation metrics

	Accuracy	Precision	Recall	F-score	MCC
Gauss NB	0.739	0.410	0.605	0.48	0.333
Multi NB	0.625	0.298	0.605	0.4	0.193
SVM Linear	0.739	0.4	0.526	0.454	0.291
SVM RBF	0.798	0.6	0.07	0.139	0.162
SVM Sigmoid	0.793	**1.0**	0.0	0.0	0.0
Decision Tree	0.717	0.325	0.342	0.333	0.154
Neural Network	0.793	**1.0**	0.0	0.0	0.0
AdaBoost	0.798	0.510	**0.631**	**0.564**	**0.440**
Randm Forest	**0.826**	0.636	0.054	0.466	0.391

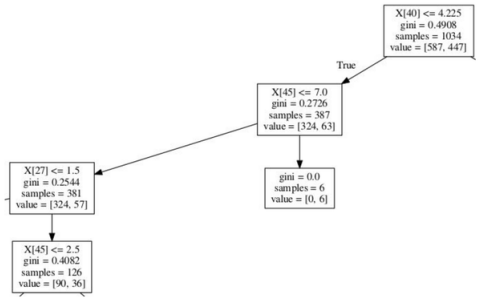
Note that no one classifier has the best performance according to all evaluation measures. For instance, Random Forest method ranked one in terms of accuracy, but its recall is one of the lowest. This means that in analyzing the performance of the algorithms using these metrics, it is crucial that researchers identify which metrics provide us with information that is most relevant to the problem at hand.

An important fact to recall is that the data set provided is unbalanced. In medicinal practice and the data set provided, positive examples(in this case, recovery) are rare. Thus, always outputting a "No"(prediction$(x) = 0, \forall x$) will actually result in somewhat high accuracy. For instance, if there are 8 Nos and 2 Yeses, the accuracy is 80%. However, this accuracy is not a good measure of the performance of the algorithm. It does not detect any true positive case.

In an unbalanced data set, it would help to think about how many of the few positive cases are properly detected. This is the idea behind recall, as shown in an equation earlier in this report. More

4.3. Results: Decision Trees

In addition to maximizing our accuracy, we also wanted to see if we could produce results containing biological significance. One algorithm that provides a happy medium between both algorithmic and biological approaches is decision trees, because we can retroactively analyze the tree splits to gain a clearer sense of its decision making. The entire decision tree is too large to feasibly reproduce, but does contains interesting splits, the most interesting being the two lefthand splits starting at the root.



The first split occurred based on the patients viral load (HIV virus particles in a milliliter of your blood). That patients with a high viral load or in other words, severe cases of HIV, were unlikely to recover makes intuitive sense and we were encouraged that the machine recognized and prioritized this. This finding is consistent with the Langford et al paper which found strong correlation between patient diagnostics such as CD4 and viral count and overall HIV progression.

After splitting on viral load, the decision tree split on feature index 45, which corresponded to column 187 in the PR sequence. We were encouraged to see that for certain bases, after just the second split, the tree could make a unanimous decision. To verify this, we consulted our MSA and found that indeed, all patients who had certain bases within column 187 did not recover.

However, as we progress down the tree, the biological significance of our splits become much less clearly defined. Due to time constraints, we were unable to consult with

external biological experts, and mainly relied on data we already had i.e. our MSA and k-mer outputs to verify significance. However, even if we cannot verify biological significance, the splits still gave us interesting hints on how we might improve future results. For example, after splitting on column 187 of the MSA, the next split on the left hand side was index 27, which corresponds to a k-mer sequence (it would helpful to get the 9-mer). While we are unsure about the genetic ramifications of [9-mer sequence], it provides us insight that the most frequent 9-mers may not be the most relevant as index 27 was the 28th most frequent 9-mer. Potentially, we might want to incorporate least frequence 9-mer as well as random 9-mers as based on the tree splits, pure frequency does not appear to highly predictive.

5. Conclusions & Future Directions

6. References

7. Appendix

Acknowledgments

We would like to thank Professor Ameet Soni, who has assisted and provided guidance throughout all facets of this project, from data collection to algorithm selection. We would also like to thank Kaggle for gathering and assembling the HIV progression data.

References

JC Hendriks, GF Medley, SH Heisterkamp et al. Short-term predictions of hiv prevalence and aids incidence. *Epidemiology and Infection*, 1992.

Thorsteinn Rgnvaldsson, Liwen You and Garwicz, Daniel. State of the art prediction of hiv-1 protease cleavage sites. *Bioinformatics*, 2015.