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 one's genetic underpinning. Given patient genomic data, we propose utilizing bioinformatic techniques such as multiple sequence alignment(MSA) and k-mer 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 an 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 virus' genetic sequences and a few other pieces of information such as a patient's viral load and CD4 count at the beginning of therapy.

This problem can be framed as a mapping; given a set of features as input, we want to output the most likely outcome. 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 k-mer. 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 to use as features. In the event that the data proved unconducive to MSA, we also implemented k-mer which does not require sequences to be aligned. From k = 3 to k = 7, we utilized k-mer 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 six unique classifiers (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 k-mer feature vectors did not enhance predictive accuracy. Using this metrics, we also found that AdaBoost was the most predictive machine learning classifier followed closely by linear SVM. Lastly, due to imbalances within our training data, we found metrics such as confusion matrices to be much more insightful than traditional accuracy.

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 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, the ability of HIV

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

virions to replicate and infect additional cells is disrupted. 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.

After analyzing how mutations to various sites within the protein sequence affect HIV protease potency, Rognvaldsson et al discovered that a linear SVM with standard orthogonal encoding is the best predictor across all data sets (Rognvaldsson 2015). In our experiments, we hope to build upon their work. Within their experiments they focused heavily on solely training SVMs and compared them across two online predictors, HIVcleave and PROSPER. Instead, we aim to compare SVMs against a wider range 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 k-mer.

2.2. Biological Indicators

In addition to building a robust model, we hope to produce helpful references that can explain whether our model's predictive power is biologically sound. Biologists Langford et al 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 (Langford 2007). Later, 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) (Poorolajai 2015). 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

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 (Korenromp 2009). 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 (Carvajal-Rodrguez 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 (Carvajal-Rodrguez 2007).

At times, biology and bioinformatics can appear disjoint as biologists generally prefer 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, or 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. A few data entries were incomplete; in this case, we disregarded them completely.

The testing set also did not contain ground truth labels, as they were intended for a competition entry. In this case, we used them to perform the MSA and nothing more. In the end, we were left with a set of 920 complete data points in this project taken from the training set, which we split 4:1 for validation.

Finally, viral load and CD4 count were already discretized variables and thus required no processing; we simply added them as features in their current state. Mainly, our task was to refine the RT and PR sequences as they were both far too long to feasibly use.

3.2. Data Processing: MSA

Not only can nucleotide sequences be hundreds of characters long, many sections of the sequences are homogeneous 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 confident that these disparities were biologically significant, not products of random offset.

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 minimum entropy on each of the result columns and ranked them based on genetic disparity. The entropy of an aligned column is defined as follows:

$$E(X) = -\sum_{i=1}^{k} P(x_i) \log P(x_i)$$

where X is an alignment of a column with constituents $x_1, x_2, ..., x_k$, and $P(x_i)$ is defined as the number of characters x_i divided by the length of the column. Thus, an absolutely homogeneous column will have entropy 0, while a column with 1:1 split will have entropy 1.

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 a significant concern given that we had less than one thousand training cases.

3.3. Data Processing: k-mer

However, recognizing that MSA has 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 sequence, and a length, k, as input, and then returns a count of all unique sub-strands of length k. 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, as previously mentioned, since k-mer only operates within the context of an individual sequence, we do not have to deal with any issues related to alignment.

To implement k-mer, we installed k-mer, 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 7-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 following parameters shown in Table 1.

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

To deal with the imbalance between negative and positive examples, we sampled each positive data in the training set $w = 3$ times. With respect to minimum entropy, we chose to use the top 40 columns with regard to variation.

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

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

3.6. Classification

In total, we utilized 6 classifiers: 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

Support vector machines were of particular interst, as Rognvaldsson et al reported that SVM with linear kernel perfomed best (Rognvaldsson 2015). The idea behind SVMs can be sumamrized as transforming data vectors from higher dimensions to find decision boundaries that might not exists in the original dimensions. Although the computation required to perform this transformation is often costly, a technique called kernel trick allows us to quickly and efficiently compute the distance between two vectors in the desired space, rendering this method effective. Given that our feature contains 122 dimensions, we believe that it dimension based transformations could potentially be utilized to great effect. We eventually applied three separate mappings to the original data: linear, radial basis (rbf), and sigmoid.

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 necessarily 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
Decision Tree	Default Parameters
Neural Network	Architecture and training parameters included in Appendix Figure C
AdaBoost	100 base learners (decision trees)
Random Forest	100 base learners (decision trees)

3.10. Analysis: Cross validation

To evaluate the performance of our algorithms, we split our training data set into a training and a test set, using a 4:1 ratio respectively. This allows us to train our classifiers and measure their performance using "novel" data, mitigating the potential of overfitting. It should also be noted that in the training phase, the ratio of training:testing set is different since the weighting of positive examples is only applied to the training set. This reduces the probability that our training set correlates with our testing set and increases our chances of obtaining more informative results.

3.11. Analysis: Confusion Matrices

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.

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

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

3.12. Analysis: F-Score

We then utilized precision and recall to obtain an F-score, which provides us with a more holistic alternative to accuracy. The F-Measure is computed by taking a weighted average of both precision and recall, and takes on a value between 0 and 1. Mathematically,

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

Specifically, 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 ability to be correct when labeling a positive. This is important particularly in unbalanced data sets such as ours when positives are less common than negatives and/or a false positive needs to be avoided. In the case of HIV progression, a false positive would be particularly damaging as it would classify a patient whose condition is expected to worsen as one who is expected to recover, potentially dissuading the patient from pursuing necessary treatment. Thus, while accuracy is important, other traits such as precision should also be prioritized.

3.13. 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. It is calculated 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

Table 4 displays the evaluative results for each classifier. For each metric, the cell(classifier) with highest value is emboldened. Table 4 excludes other helpful metrics such as confusion matrices and decision trees which are elaborated upon in more detail in 4.2 and 4.3.

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	**0.815**	0.583	0.368	0.451	0.360

4.1. Results: Evaluation Metrics

Note that no one classifier holds the best performance across all evaluation metrics. For instance, Random Forest method ranked first in terms of accuracy, but its recall is one of the lowest. This means that in analyzing the performance of the algorithms, in addition to accuracy, it is crucial that researchers consider which additional metrics are most relevant to the problem at hand.

For example, in our study, the data set provided is unbalanced. Likewise in the real world, positive examples(in this case, recovery from HIV) 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 since the algorithm 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 the equation provided

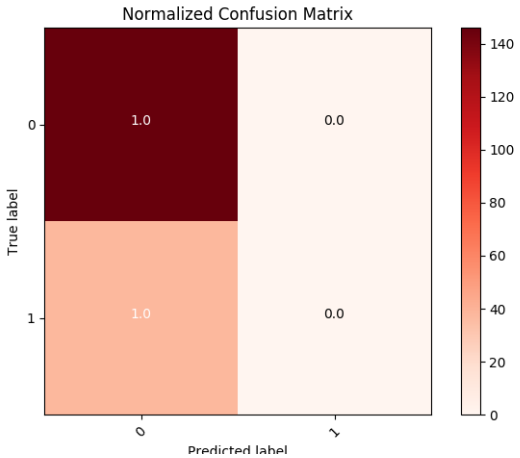
earlier in this report. Thus, it makes sense to treat recall, F-score and MCC more importantly than accuracy and precision. Otherwise we would consider the unweighted neural network classifier, which only outputs positive predictions, the third best classifier.

Therefore, we conclude that recall is the most useful metric to use since it captures how much of the total positive examples a classifier has correctly predicted. However, note that recall also could be easily improved to 1.0 if a classifier uniformly predicts 1. Therefore, while keeping the rank of precision values in mind, it is necessary that we somehow combine the various statistics we have into a single, comparable value (F-score and MCC). Overall, AdaBoost was observed to perform the best in measures of recall, F-score and MCC, which is why we consider AdaBoost as the best classifier, despite it not being the most accurate.

4.2. Results: Confusion Matrices

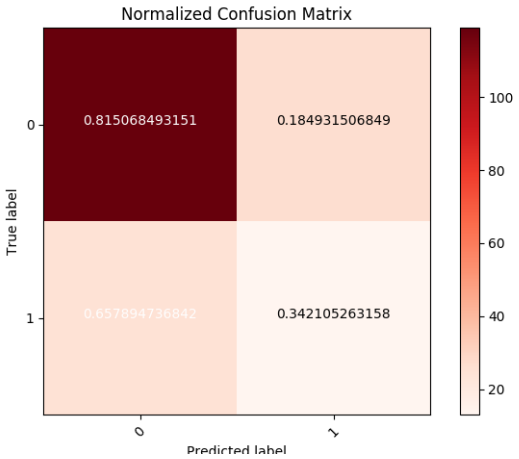
In addition, analyzing the confusing matrix yielded helpful insights as to how our classifiers were getting predictions right. For instance, the confusion matrix for neural networks confirmed our intuition that due to the heavy imbalance of progression cases in our training set, certain classifiers attempted to gain accuracy by classifying all cases under one label (Figure 1). Each answer turned out to either be a true or false positive meaning that the classifer only output one answer (positive).

Figure 1. Normalized Confusion Matrix of Neutral Network Classification Result



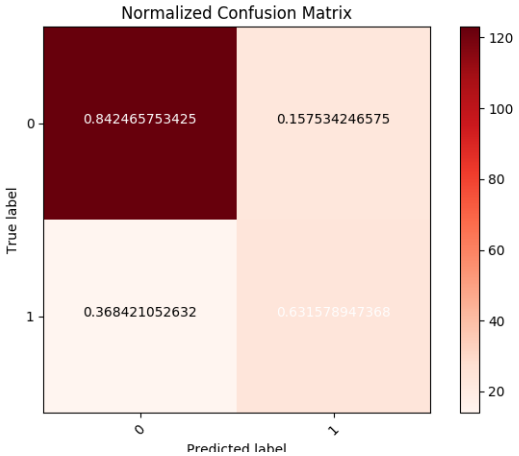
From this, we could also see how well our data processing improved our results. After performing data augmentation and weighting regression cases to balance our training set, the same neural network algorithm produced the following confusion matrix (Figure 2).

Figure 2. Normalized Confusion Matrix of Decision Tree Classification Result



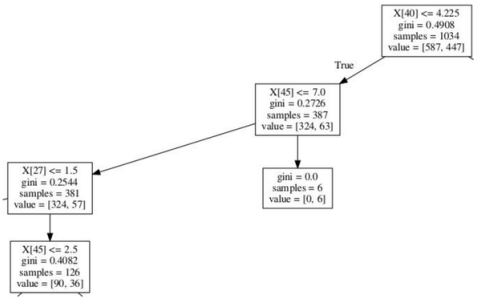
While the overall accuracy was slightly less in this case, we feel much more confident in the latter neural network's ability to generalize to novel data. This is a helpful example as to why accuracy is not the end all be all. We then analyzed the matrix of our best performer, AdaBoost (Figure 3). Based on the matrix, we conclude the best classifiers such as AdaBoost were able to discern relevant patterns within the data noise, as the composition of false negatives and false positives was much more balanced than that of Figure 2 which suggests that it is not overcompensating to accommodate the training set. It also suggests that AdaBoost discovered certain rules of thumb that could be used to methodically determine outcomes. Despite this, we still feel that some aspect of the training set skew is nonetheless present as the prevalence of false negatives relative to false positives is still higher than we'd like it to be.

Figure 3. Normalized Confusion Matrix of AdaBoost Classification Result



4.3. Results: Decision Trees

Finally, we also wanted to see if we could produce results of 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 (a larger subtree is provided for reference in Appendix Figure C).



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. While we are unsure about the subsequence's genetic ramifications, it provides us insight that the most frequent 7-mers may not be the most relevant as index 27 was the 28th most frequent 7-mer. In the future, we might want to incorporate least frequent 7-mers as well as random 7-mers as feature vectors; based on the tree splits, pure frequency does not appear to highly predictive.

4.4. Additional Remarks

We were not surprised to see linear SVMs perform relatively well as prior work by Rognvaldsson et al demonstrated that linear SVMs were the best predictor relative to other bioinformatic tools. However, unlike Rognvaldsson et al, we found that AdaBoost, not linear SVMs, were the overall best performer. We are curious if AdaBoost would also work just as effectively on the data set utilized by Rognvaldsson et al. and see this as a logical future point of study.

In addition, our results support the idea that there are several key sites within the HIV genomic sequence that influence the virus' ability to progress. The splits from our decision tree show that in certain cases, patient outcome can be determined by as few factors as viral load and the base within one site, and with 100 percent accuracy (at least with respect to our training set). The presence of certain subsequences also appears to be correlated with severity, though it is not an extremely strong proxy, evidenced by how our machine learning algorithms accounted for k-mer features.

Finally, our results from performing data augmentation and weighting remission cases lend credence to the fact that our training set is improperly balanced. This makes it difficult to attribute mistakes to deficiencies within our methods as opposed to deficiencies within the data set. Quick analysis of the confusion matrix pre and post data augmentation for neural networks and even effective classifiers such as AdaBoost clearly show that our classifiers were compensating for the skew by weighting towards false negatives when unsure. We believe that further testing on more unbiased data is necessary before a definitive conclusion on each classifier's effectiveness can be reached.

5. Conclusions & Future Directions

We find that machine learning algorithms definitely have the potential to serve as robust HIV diagnostic tools. Even weak classifiers have demonstrated the ability to recognize relevant patterns within the noise and improvements upon weak classifiers such as AdaBoost have shown particular improvement. That being said, there is still a great deal of ground left to cover as even the best algorithm could only accurately diagnose four out of every five cases, a terrible percentage by modern diagnostic standards. However, being that our study only spanned several weeks, it is reasonable to assume that even utilizing only our current methods, we can expect better results given more time to adjust pa-

rameters, and better tailor the training data.

There are several improvements that we believe could lead to more robust results. First, methods such as our decision trees have shown that the most frequent 7-mers may not necessarily be the most predictive subsequences. It is not without the realm of the possibility that the most predictive subsequences are among the least common or even tucked away somewhere in the middle. It would be interesting to run versions utilizing k-mer where we probabilistically choose which 7-mers to use as features, as opposed to only choosing the most frequent twenty.

In addition, we conducted our MSA using both the training data and test data in an effort to create meaningfully aligned columns. However, when performing minimum entropy, we excluded the test set that we aligned since we did not have answers for the test set. We are concerned that because the training set is heavily imbalanced while the test set is not, given that they are both relatively equal in size, the MSA is not as optimal as it could be. In our opinion, it would be better to perform an original MSA on just the training set and then perform a new MSA every time we get a new test case. Over time, this might shift the minimum entropy columns, so we would have to implement more dynamical analytic tools so our program still picks the correct columns.

Furthermore, a great deal of the training set contained symbols that were defined probabilistically, such as Y, which meant that it was either C or T. Because there isn't a great way to discretize nucleotide bases, we ended up assigning each base a unique integer, which made our base dictionary quite large. We definitely believe that our results would improved if we could ascertain which character these probabilistic bases were representing. At the very least, it would be helpful to consult with experts to determine the relative likelihood of each substitution i.e. does Y mean that it is C and T both occur 50 percent of the time or is it a 90/10 split?

Moreover, we would be interested in seeing if ensemble methods such as boosting could enhance our results. Previous empirical literature in other fields has demonstrated that certain classifiers may be more adept at classifying different subsets of data and that weighting can utilized to grant select classifiers more credence in various cases. We assumed that all classifiers were uniform in their ability to classify each individual case and similar to AdaBoost, our results may benefit if we partition the data into clusters, test classifier performance amongst the different clusters, and then choose the best classifier for each cluster.

Finally, we would be interested in performing our experiment given access to more features related to the patient. As many related works show, in addition to viral genetics, patient genetics such as sex and age also influence viral progression. We are concerned there may potentially be a performance bottleneck with access to only the patient's viral sequence as the same mutation may affect patients across different demographics differently.

6. Appendix

Figure A: Classifier Overview (excluding SVMs and AdaBoost which were mentioned earlier)

- Naive Bayes: Naive Bayes starts with a prior and updates given new information. This is particularly suitable to our experiment as each feature represents new data that the machine can use to update it's classification probabilities.
- Decision Tree: Decision trees classify by splitting across each feature. Visualizing the splits helped us identify which features were the most relevant.
- Neural Network: Neural networks are epitomized by their deep learning capacity; it is usually expected that a sufficiently capable neural network will learn different levels of representations of the data, and use such abstractions to "reason" about which classification is best. Thus, researchers might use neural networks in the hopes that it might be able to capture various long-term dependencies and higher-level structures arising from neural networks.
- Random Forest: Random forest works by sampling subsets of the data to train many iterations of a given classifier (in this case decision trees). Although there is no one algorithm that excels at all problems, ensemble methods like random forest are generally improve upon weak classifier results. In theory, it is expected that combining predictions from multiple weak classifiers will reduce overfitting.

Figure B: Neural network Parameters and Architecture

The following figure contains a summary of our neural network's architecture and parameters.

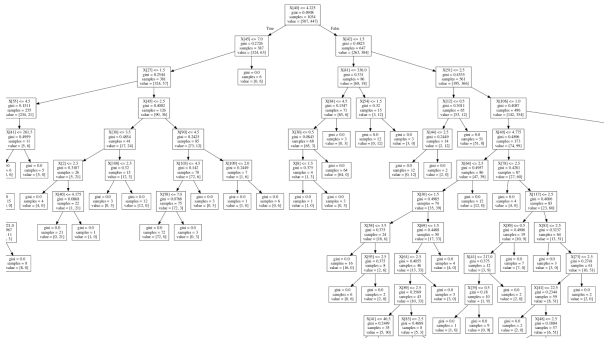
Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 32)	3936
activation_1 (Activation)	(None, 32)	0
dropout_1 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 32)	1056
activation_2 (Activation)	(None, 32)	0
dropout_2 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 2)	66
activation_3 (Activation)	(None, 2)	0

Total params: 5,058
Trainable params: 5,058
Non-trainable params: 0

The first two hidden layers contain 32 nodes each and utilize a relu activation function(rectifier linear unit). Dropouts of both 0.2 and 0.5 are implemented in an effort to reduce overfitting and achieve the effects of ensemble learning. The output layer converts the soft output into either 0 or 1 by taking the soft max of two values in the output vector. For a visual reference, see below:

Link to the git repository containing the png file describing the architecture of the network

Figure C: This is a larger subtree picture of decision trees that can be found in the google doc



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

Carvajal-Rodriguez. The importance of bio-computational tools for predicting hiv drug resistance. *Recent Pat DNA Gene Seq.*, 2007.

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

Kleinberg, Jon, Ludwig, Jens, Mullainathan, Sendhil, and Obermeyer, Ziad. Prediction policy problems. *Am Econ Rev .*, 2015.

Korenromp, E., Williams, B., Schmid, G., and Dye, C. Clinical prognostic value of rna viral load and cd4 cell counts during untreated hiv-1 infectiona quantitative review. *PLOS*, 2009.

Langford, SE, Ananworanich, J, and Cooper, DA. Predictors of disease progression in hiv infection: a review. *AIDS Res Ther.*, 2007.

Poorolajal, J, Molaeipoor, L, Mohraz, M, Mahjub, M, Ardekani, MT, Mirzapour, P, and Golchehregan, H. Predictors of progression to aids and mortality post-hiv infection: a long-term retrospective cohort study. *AIDS Care*, 2015.

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

Sajda, Paul. Machine learning for detection and diagnosis of disease. *The Annual Review of Biomedical Engineering*, 2006.

Shankaracharya, Odedra, Devang, Samanta, Subir, , and Vidyarthi, Ambarish S. Computational intelligence in early diabetes diagnosis: A review. *The Review of Diabetic Studies*, 2010.

Statnikov, Alexander, Aliferis, Constantin F., Tsamardinos, Ioannis, Hardin, Douglas, and Levy, Shawn. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, 2004.