THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF BIOCHEMISTRY AND MOLECULAR BIOLOGY

RANDOM FOREST CLASSIFICATION IN COPY NUMBER VARIATION DISCOVERY

GOPAL JAYAKAR
SPRING 2020

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Biomedical Engineering
with honors in Biochemistry and Molecular Biology

Reviewed and approved* by the following:

Santhosh Girirajan
Professor of Genomics
Honors Advisor & Thesis Advisor

Shaun Mahony
Professor of Biochemistry and Molecular Biology
Honors Reader

* Electronic signatures are on file in the Schreyer Honors College.

ABSTRACT

As sequencing technologies and machine learning methods advance, the potential to diagnose genetic diseases and conditions increases. The leading genetic sequencing platforms, Illumina included, all generate their sequencing output in the form of short genetic sequences on the order of tens to hundreds of base pairs called "reads". When generating a sequence of the entire human genome, specialized software is used to stitch shorter reads together until a large contiguous genetic sequence can be output. The current approach to genetic sequence elucidation has several weaknesses, including difficulty detecting a type of genetic anomaly called a Copy Number Variation (CNV).

CNVs are either duplications or deletions in the genome and are larger than Single Nucleotide Polymorphisms (SNP). CNVs have been implicated in the etiology of a wide range of conditions, including Intellectual Disability (ID), Autism, and Schizophrenia. The listed conditions here are all neurodevelopmental, CNVs have the potential to affect any area of health. Accurate identification of CNVs from available sequencing data has the allure of providing diagnostic potential (Clancy, 2008).

Currently available CNV identification algorithms have large false positive rates, potentially suggesting incorrect diagnoses. The individual algorithms additionally display little concordance which makes correct CNV determination (termed a "CNV call") difficult without an external source of validation. This project attempted to create a higher quality CNV calling algorithm by first polling several extant CNV algorithms, comparing and combining their outputs, and using various quality-control metrics to generate a summary of their results.

These results are used as features in a random forest machine learning model. Machine learning is a process by which computers can be trained to perform arbitrary tasks, and the random forest model is a machine learning approach designed to assign categorical values to each input using a gold standard as a reference to reinforce correct predictions. The “gold standard” used for comparison was microarray SNP data. In this instance, the random forest model is assigned the task of deciding whether the input is a duplication, deletion, or there is no CNV. Using this approach, a higher quantity of CNV calls with greater precision and recall was recovered than any of the individual algorithms could produce. With further refinement, the methods used in developing this algorithm could be used in medical practice to diagnose a variety of conditions with genetic origins.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank Vijay Kumar, Matt Jensen, Neil Kelkar, and Santhosh Girirajan for their leadership, guidance, collaboration, and advice in the completion of this project.

Additionally, I would like to express my gratitude to Shaun Mahony for agreeing to read and provide comments on this thesis project.

Lastly, I would like to thank Andrew Polyak for his mentorship as I began my work in the Girirajan Lab.

This work was supported by R01-MH107431, R01- 472 GM121907, SFARI Pilot Grant 399894, and resources from the Huck Institutes of the Life Sciences 473 to Santhosh Girirajan. This work was funded partly by the Big Data to Knowledge (BD2K) pre-doctoral training 474 program (T32LM012415) from the National Institutes of Health to Vijay Kumar Pounraja. The funding bodies had no 475 role in data collection, analysis, and interpretation.

Introduction

Machine learning has grown to define the field of computation in recent years, and is responsible for advances in various diagnostic tools. In general, machine learning is an algorithmic process by which a computer program can approximate learning by making many small adjustments to its own code, and preserving those modifications that best allow it to perform the task provided. Due to the high computational power of machines, and the ever-growing quantities available to modern bioinformaticians, machine learning continues to grow in healthcare. Recent advances have increased machine learning's share in such fields as health monitoring, follow-up appointment optimization, and diagnostic tool improvements (Bhardwaj, Nambiar, & Dutta, 2017).

Many types of machine learning exist, each with their particular strengths. This brief review will focus primarily on classifiers, which are algorithms trained to ascribe a label to its input. At the most fundamental level, the two types of machine learning are "supervised" and "unsupervised", and are utilized based upon whether or not the user has access to the "ground truth". For example, a scientist interested in building a machine-learning model which uses dental x-rays to diagnose cavities would provide a database of x-rays to the machine learning model, with labels indicating whether a dentist who examined the x-ray believed there was a cavity in the image. In this instance, the dentist would be providing the "ground truth", which can also be thought of as the "correct" prediction. The ground truth and the given images or other datapoints are called "training data".

Supervised learning classifiers can take in both the given data and the ground truth, and use the expected outcome to decide which factors should be considered in generating a classification. In the case of a dental x-ray, such factors could include thin layers in certain regions of the tooth or abnormalities in the tooth profile. For supervised learning, an accurate ground truth is extremely important, because inaccuracies in the training data will result in inaccurate predictions. These predictions can be tested within the machine learning model itself; by segmenting the given data (with the provided ground truth) into two categories; training data and test data. The model will train itself using the ground truth and training data, and determine its own accuracy by comparing the predictions it would make on the test data to the corresponding labels. Making small adjustments each iteration, a supervised learning program can slowly decrease the number of misclassified inputs until it has successfully “learned” to perform the task.

The other major type of machine learning is unsupervised learning, where data are given to a machine learning program without indication of the ground truth. Instead of provided labels, like “cavities” or “no cavities”, the machine learning algorithm will attempt to generate its own categories, without traditional labels. Because the classification is blind and no ground truth is provided, the generated categories are not meaningful without further analysis. It is possible that the machine learning model will discover relations between the data scientists had not discovered before, making unsupervised learning an interesting tool for analysis of large, less-interrogated data.

Machine learning, while powerful, has certain limitations that cannot be exceeded using current methodologies, depending on the particular algorithm implemented. The most relevant limitation within science is perhaps machine learning’s lack of explanatory power. While excellent at identifying correlations within data, machine learning cannot explain why two variables may be related. Judea Pearl posits a three-leveled hierarchy of causal reasoning; association is the most basic kind, intervention at the intermediate level, and counterfactuals at the highest (Pearl, 2018).

The most basic level of causal reasoning is easily within reach of machine learning. Association between two variables, like a particular symptom and disease likelihood, are the primary target of many modern machine learning methods. At the second level, intervening, Pearl suggests that a true AI would be able to ask hypotheticals. For example, if performing a certain surgery would alleviate a specific set of symptoms. Given robust training, machine learning methods may be able to help answer some of these questions, but extrapolation far beyond the range covered by the training dataset will lead to nonsense results. Counterfactuals are at the highest level, and ask “why”. For example, why might a certain medication cause a particular side-effect. At the current moment, Pearl argues that these questions are currently off-limits for machine learning. Within the frame work of biomedical research, establishing causality is highly desirable for a full understanding of the etiology of disease.

There are a wide variety of machine learning classifiers, which all differ both in the methods by which they classify inputs and the amount of information they can provide about their internal logic. Some of these methods include Linear Classifiers, Support Vector Machines (SVM), Decision Trees, and Neural Networks. Of these, the linear classifier is perhaps the most basic.

Linear classification models are created from datasets that have many samples, each with a set of associated numerical variables and divided into two or more classifications (Mitchell, 2017). Linear classifiers, given two input variables per sample, essentially plot the samples on an x-y plane and assume that there exists a line that can divide the two populations. Before running the program, the user can determine the form of the line; for example, logistic regression is a form of linear classifier that assumes the line dividing the items of the different classifications is of the form $y = \frac{1}{1+e^{-x}}$. Additionally, a cost function must be defined. A cost function indicates the performance of the model based upon how accurately it classifies the samples provided to it. By making small modifications in the various coefficients and powers and retesting every modification using the cost function, the dividing line will converge to the point at which it most accurately separates the classes. These methods can be extrapolated to more categories and higher variable counts, but have the advantage of always yielding an equation upon their conclusion, making their decision-making process highly transparent. SVMs take an extremely similar approach, but instead maps the variables into higher dimensional space and creates hyperplane boundaries that can take complex shapes (Asa, Horn, Siegelmann, & Vapnik, 2001).

Decision trees encompass a different approach to classification than Linear Classifiers or SVM in that they do not attempt to represent the classes in any space (Horning). Instead, decision trees are recursive structures that begin with a root note and many child nodes. Each node represents a decision, which could be as simple as whether or not a specific variable exceeds a threshold (e.g. “age” > 50 years). Each node will branch either into other nodes or a classification, offering a clear path to classification of a novel input given all the associated variables. Many approaches to generating and optimizing these structures exist, and they sometimes do include optimization criteria other than overall accuracy. Some decision trees have additional target function that reduce the number of nodes or minimize the depth of the tree so that a human examiner might be able to understand the tree's decision-making process.

Creating a truly optimized tree is an extremely difficult task, which is why using a Forest might be advantageous. The Random Forest Classifier creates a wide variety of decision trees, each with unique architectures (Horning). When encountering a new sample, Random Forest classifiers poll each decision tree within it and use the results to assess the probability the sample lies in each of the categories. While technically easily visible, the decision-making process within a Random Forest classifier is unintelligible to humans because a large number of diverse trees are employed in this process. The benefits of using a random forest are many. The unique architecture of the various trees ensures that they are de-correlated, which makes the forest noisy, but the final classification is performed by polling the trees or averaging their results. In this manner, classification can be made using unbiased models that can capture extremely intricate correlations between variables (Hastie, Trevor, Tibshirani, Robert, Friedman, 2009).

Although the specific reasoning behind classification decisions in Random Forest Classifiers is opaque, variable importance data can be recovered from these programs. Each node in each tree ascribes some amount of importance to each variable, and when aggregated, a total variable importance score can be found. Other benefits of the Random Forest Model include its ability to provide accurate predictions even when supplied with a large number of variables assuming a large number of relevant variables (Segal, 2004) and that Random Forest Classifiers, while they can potentially overfit to certain datasets, are unlikely to do so (Hastie, Trevor, Tibshirani, Robert, Friedman, 2009). The combination of all of these advantages make Random

Forest Classifiers extremely useful not only in terms of their ability to classify novel elements properly, but also in that they are fairly forgiving on the user even if the ideal set of features is not properly identified, so long as few relevant features are excluded.

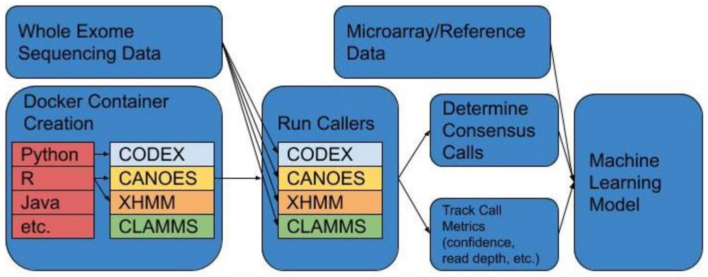
The robust nature of Random Forest Classifiers makes them a powerful tool in the analysis of complex biological systems, where the most relevant variables can frequently be difficult to identify. This project uses Random Forest Classification on the outputs of existing whole exome sequencing (WES) based CNV calling algorithms to make calls with higher levels of confidence than was previously possible. Increased levels of confidence are made possible not only because of the classifier used, but also because of the benefits of ensemble approaches. An ensemble approach is where many different methods of making a prediction are used alongside one-another, typically where each methods has its own particular advantage or niche; similar ensemble methods have been used to call CNVs from SNP array data (Zhang, et al., 2019). The machine learning method will then combine and refine the results of each individual prediction method to generate a prediction with higher confidence. In the case of CN-Learn, this is implemented by using the results of four different WES-based callers as well as passing along statistics like coverage depth and GC-content from the initial sequencing run.

My personal contributions to this project included developing pipelines and generating CNV calls for 2 of the 4 callers. This work also included discussions and development of the best methods by which consensus calls could be generated. Additionally, I developed the Dockerfile that outlines what and how to package all the required software tools into a Docker container, allowing for increased portability of the software. Lastly, I aided in testing and bugfixes for the project.

Materials and Methods

All work was performed on computational servers at Penn State. For the purpose of downloading CNLearn, a Github page was created: https://github.com/girirajanlab/CN_Learn. The basic pipeline for CNLearn is summarized in the figure below. After all the individual callers are installed, they are individually used to call CNVs from the available patient data. Those calls are then overlapped with some tolerance to determine areas of potential agreement, then provided to the machine learning model.

Figure 1: CNLearn Pipeline



Whole Exome Sequencing Data
Microarray/Reference Data
Docker Container Creation
Python
R
Java
etc.
CODEX
CANOES
XHMM
CLAMMS
Run Callers
CODEX
CANOES
XHMM
CLAMMS
Determine Consensus Calls
Track Call Metrics (confidence, read depth, etc.)
Machine Learning Model

A Dockerfile present on the github page can be used to both create the required filesystem as well as install the requisite supporting programs needed for both the machine learning model and the supporting CNV calling algorithms to function properly. It is based on rocker, a dockerfile designed to provide easy access to various R builds—in this case, R 3.4.4. The Dockerfile will automatically download and install various R packages, including sqldf, devtools, and Rcpp, as well as bioconductor functions and an install of CODEX. In building the docker container, the dockerfile will also automatically download the contents of the github page, including the filesystem.

Other important software downloaded include python3.7.3 and associated packages, like scipy (the machine learning software). samtools and bedtools (required for file preprocessing). Meanwhile, the other CNV calling algorithms are installed: XHMM, CLAMMS, and CANOES. Each of these algorithms require their own particular pre-processing steps and have their own areas of advantage over the others. Each algorithm additionally follows its own process for generating read depth, which can be examined in their respective program files.

CODEX (Jiang, Oldridge, Diskin, & Zhang, 2015)

CODEX, a CNV calling algorithm based in python, makes certain simplifying assumptions about the read depth distribution in the data in order to generate calls. The distinguishing feature of CODEX is its simultaneous approach to read depth normalization and smoothening, which compensates for some of the innate noise in Whole Exome Sequencing. By developing an expected coverage map assuming no CNVs, CODEX is able to examine the coverage regions with large deviations and determine which potential sources of noise (such as GC content) may add the most bias.

Additionally, CODEX estimates the effects of K latent variables to further reduce noise in the dataset. It is critical that the correct value of K be chosen for accurate predictions, as large K could smooth some true CNV signals. CODEX determines several potential values of K using Akaike and Bayes information criterion, but frequently uses a more conservative value of K to avoid eliminating true CNVs in favor of more noise. Ideally, this value would be chosen by identifying the elbow on a scree plot, but as the CODEX publication notes, this method is inconsistent as the plot does not always have a clear elbow. This means that the value of K is chosen somewhat arbitrarily. Further tuning this K-value is an area of potential improvement both for CODEX, and consequently, CNLearn.

When the process of normalizing is complete, CODEX seeks areas of the genome that still differ greatly from the norm out of the population of the samples provided, assuming a Poisson distribution of the underlying data. This approach means that CODEX does not need to use a reference file indicating what normal coverage in the different genomic regions may be, but also introduces a reliance of CNV calls on the other samples concurrently being processed. This may be an issue if the training set used is either extremely small or has a large population with a specific CNV.

If a large proportion of the screened individuals have low coverage at the CNV site and few samples without the CNV are introduced, CODEX may not accurately classify the CNV as a deletion because "expected" average coverage at that locus will be low. This also means that, like many other CNV callers, it is easier for CODEX to detect rare CNVs than common ones. This trend is reflected in the original CODEX publication, wherein the recall rates for rare CNVs were far better than those for commons CNVs. These proportions may change with the selected K-Value, but as recall increases, precision is likely to suffer.

XHMM (Fromer & Purcell, 2014)

Unlike CODEX, XHMM uses a more traditional approach to normalization. XHMM and several other callers use Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) to filter out noise after depth of coverage is found. CNLearn also calculates the optional GC-content and read complexity using Plink. These metrics allow XHMM to exclude regions with extremely high GC-content and regions with relatively low complexity. High GC-content greatly decreases the accuracy of read-depth estimation, and is currently an inherent limitation on NGS techniques (Chen, Liu, Yu, Chiang, & Hwang, 2013). This will affect all of the CNV callers, and even impacts the accuracy of the gold standard microarray data (Xia, 2010).

After the low-quality regions are filtered out, XHMM uses SVD to determine along which variables the read depth changes most, and eliminates them from further analysis. Similar to the K-value selection within CODEX, the elimination of these variables is intended to remove noise that does not relate to the CNV state. Those variables are zeroed out, and the remaining data is normalized. The user is expected to provide data like the "Standard deviation of DELETION z-score distribution." Fromer and Purcell provide default values for this metric and other user-provided metrics such as "Mean of DUPLICATION z-score distribution" and "Mean number of targets in a CNV call," but these default parameters have not been thoroughly investigated and will vary between sample populations. XHMM attempts to remedy this issue by using the quality control metrics calculated before to filter incorrect CNV calls.

The actual CNV calls are all done by XHMM's Hidden Markov Model (HMM). A Markov Model represents the transition between several states, each depending on the current state. For example, transitions between regular read depth to read depth indicating a duplication or deletion. A Hidden Markov Model is a system in which the current state is also unknown to the observer, useful for situation like CNV calls where even the true classification of the current state is unknown. Using the aforementioned metrics and assumptions regarding the transmission probability between the various states, the Viterbi algorithm can be used to determine which states exist at which points.

CLAMMS (Packer, et al., 2016)

CLAMMS and XHMM exhibit a large number of similarities, both in their pre-processing steps and regarding the way they perform CNV calls. CLAMMS also begins by filtering out regions with extreme GC content and poor mappability, and eliminates rather large regions surrounding the extreme GC content areas. Unlike XHMM, CLAMMS performs its normalization after binning by GC content within certain windows. This, combined with the strategy of GC filtering, reduces the amount of error introduced by GC content. To find the diploid mean and standard deviation, as well as a parameter relating to a zero copy number and a point mass flag, CLAMMS employs a mixture model fit using the expectation-maximization algorithm. This process is performed individually over each target region, which allows for finer tuning of these parameters in regions where regular coverage reasons may differ due to other uncontrollable factors. Implicitly, calculating the mean and standard deviation in this model assumes a gaussian distribution of coverage depth, distinguishing CLAMMS from the other callers.

CLAMMS uses an HMM extremely similar to the HMM employed by XHMM. The HMM model in CLAMMS additionally uses the four components calculated by the mixture model to estimate the emission probabilities between the various copy number states. This is the primary difference between the calling method for these two algorithms.

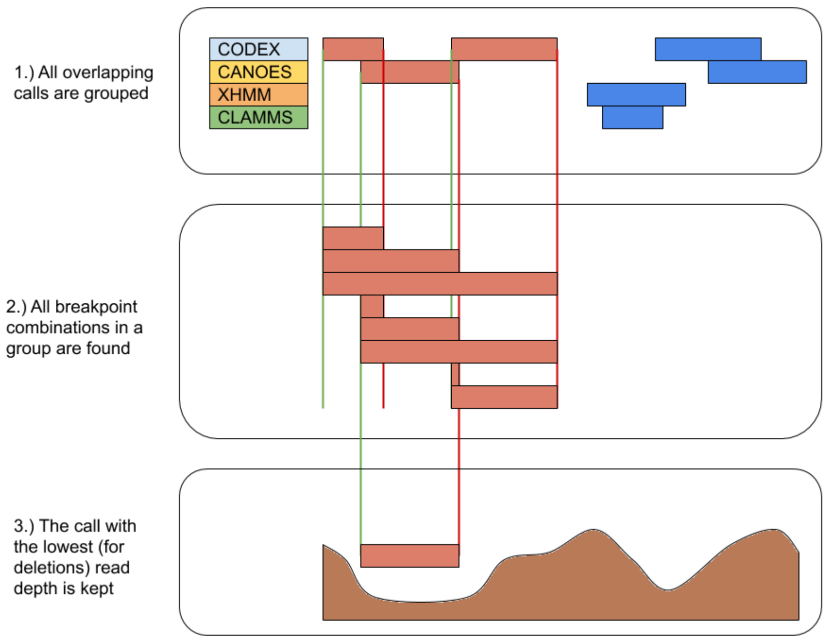
CANOES (Backenroth, et al., 2014)

The workflow in CANOES is perhaps the most similar to XHMM. Like XHMM, normalization within target regions is conducted. Instead of normalizing within target regions, CANOES elects to normalize the read count of the reference sample to match the read count of the analyzed sample. Then, read depth means and standard deviations within each target region; as is standard, these metrics are used as to calculate the emission probabilities for a Hidden Markov Model which performs the actual calling. The fundamental difference between CANOES and XHMM is its assumption that the distribution of read depths overlapping a target region follows a negative binomial distribution. Because CANOES uses its own reference files instead of relying on large batches of analyzed patients to generate its expectations for read depth, it is more robust than XHMM when analyzing fewer samples.

Developing Consensus Among Callers

Each of these four algorithms artificially segments WES probe regions differently when generating calls, and as such they have slightly different cutoff points for the same copy number event. In order to resolve this discrepancy, a three-step process was used to both merge the calls as well as attempt to discover the most accurate breakpoints for the CNV call. Step 1 involves generating groups of CNVs that overlap with one-another to any extent, as shown in Fig. 2.

Figure 2: Consensus Finding and Breakpoint Resolution


1.) All overlapping calls are grouped
CODEX
CANOES
XHMM
CLAMMS
2.) All breakpoint combinations in a group are found
3.) The call with the lowest (for deletions) read depth is kept

Once the various groups are determined, a list of all the potential start-and-stop coordinates is generated, creating a group of all the possible calls that fit any combination of the start-and-stop points. Of these possible calls, the one with the lowest read depth (for deletions) or highest read depth (for duplications) is kept and is considered the consensus call. In the case that only one caller determined a CNV happened in a region, this CNV is kept without examining read depth.

All of these calls, both the ones determined by consensus and by individual callers without support, are then classified as “True” or “False” based on overlap with the microarray data. Unlike when determining consensus, a 10% reciprocal overlap was required for classification. 10% reciprocal overlap indicates that two regions will be considered truly overlapping if the overlap size is at least 10% of the size of the smaller region. These samples, tagged as “True” or “False” and accompanied by metrics like GC score and mappability within the sample, are the inputs to the Random Forest Algorithm.

Random Forest Classifier

Python’s scikit-learn library was used to implement the random forest classifier. Interestingly, because the library was used, it is fairly easy to change CN-Learn to use either the SVM or LR strategies for classification instead of the random forest. The predictors within the random forest model are the overall concordance, GC content within the prediction, mappability, CNV size, read depth ratio, CNV type, Target Probe Count, and chromosome number, combined the proportion of the call (or consensus call) to calls made by CLAMMS, XHMM, CODEX, and CANOES. Hundreds of trees are generated, and a recommended 70% proportion of the microarray-validated dataset should be employed to train the model, while the 30% remaining is used as the test set. At this point, the classification model is constructed and can be used on unvalidated data for CNV identification.

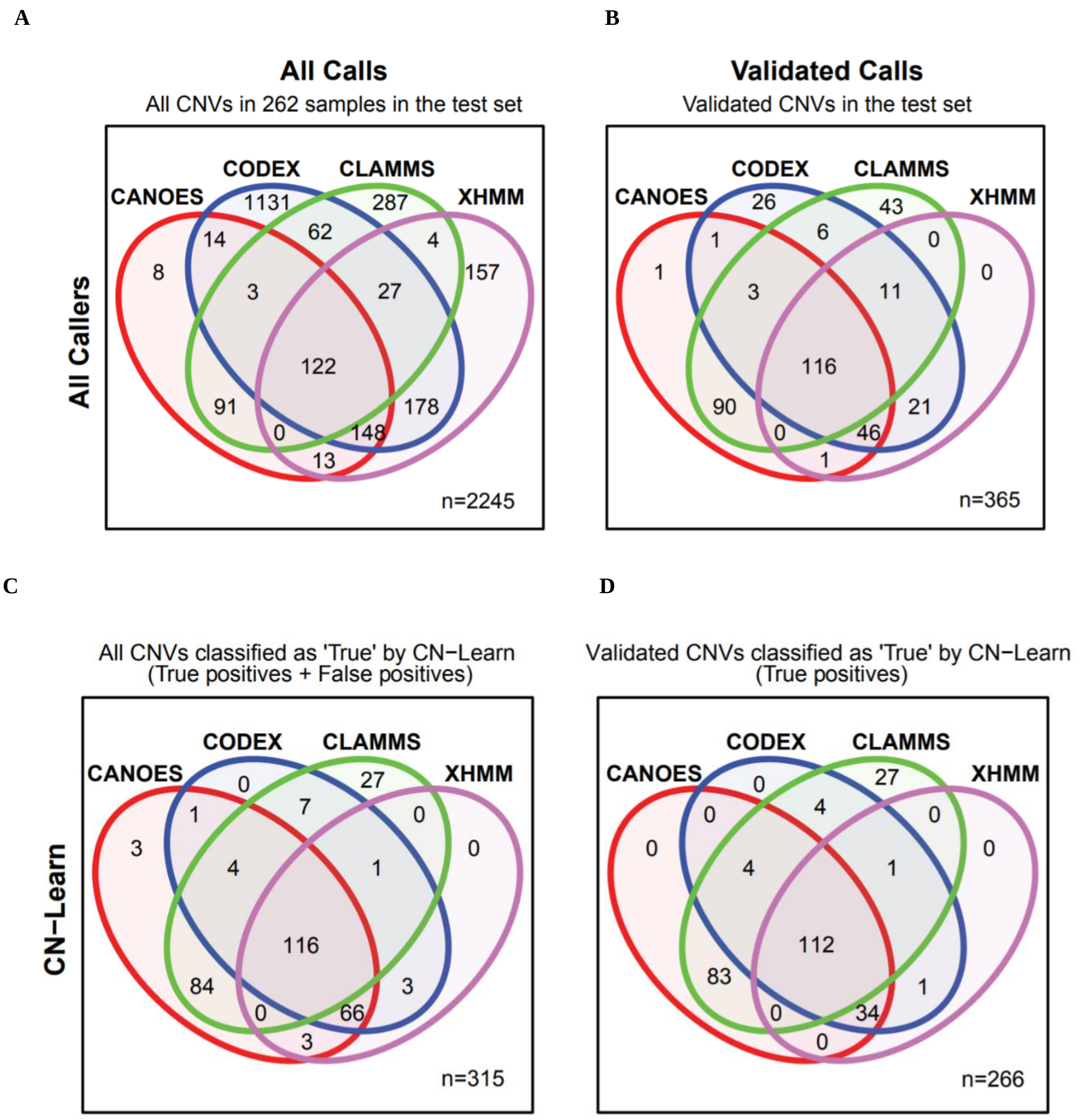
For the purposes of training the model, a set of 503 samples was acquired from the Simons Variation in Individuals Project (SVIP).   Of these 503 samples, 291 had microarray validation. As such, this was the size of the combined training and test datasets. Further experiments were performed on the data as though CLAMMS were the gold standard, but because this strays from the biological ground truth, these results will not be discussed in this paper.   Additional trial runs were performed with various training set percentages, from 10% to 70% to examine precision-recall with varying amounts of training data. In total, the four calling algorithms initially predicted 41,791 CNV events throughout all of these samples, which were then run through the aforementioned breakpoint resolution process, yielding around 30,000 CNVs events. Further analyses were run using the 291 samples with microarray validation, examining the sites where microarray probes were able to interrogate the exome.

Results

The data presented below (Fig. 3A, 3B) represent a run of CNLearn on 262 of the samples from the SVIP dataset. Of the 2245 calls made by the various CNV callers, 365 were validated by microarray. CNLearn, on the other hand, identified a set of 315 samples that it claimed to be true CNVs, of which 266 overlapped with the microarray validation. Of the various callers, CODEX lost the maximum number of calls after it was run through CNLearn, going from 1685 CNV calls to 198 CNLearn-validated calls. Of all the individual callers, CODEX also made the largest number of calls, by a factor of almost three. On the other end of the spectrum, CANOES, the algorithm with the fewest number of calls, also retained the highest percentage of its calls after running through CNLearn.

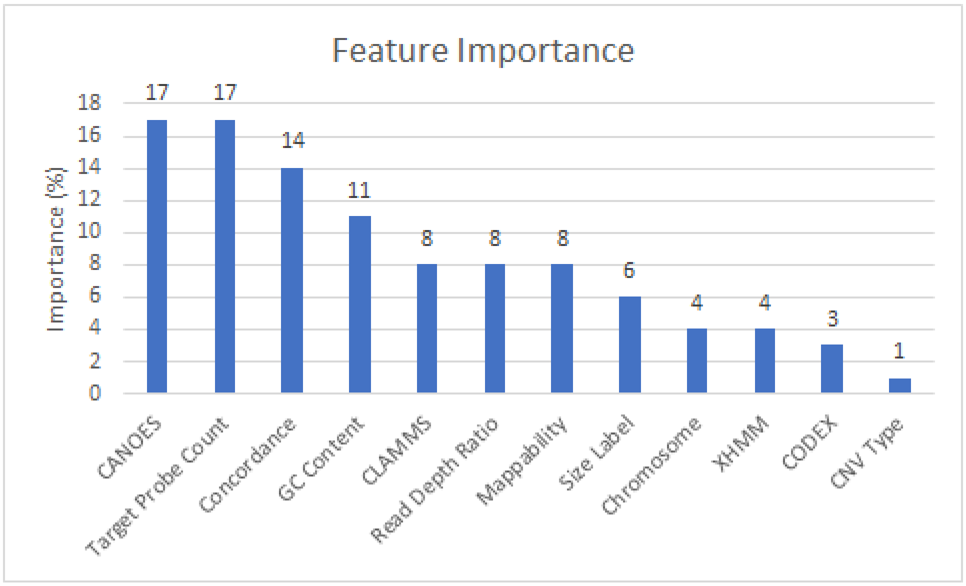
When examining the true positives (Fig. 3D), it is interesting to note that the largest value in the Venn Diagram lies at the extreme center, with a total of 112 samples. This result indicates the importance of concordance among callers in CNLearn's decision-making, a result recapitulated in Fig. 4. Indeed, only 27 calls made by a single caller, all of them belonging to CLAMMS, were considered true by CNLearn.

Figure 3: Precision/Recall and Call Number Across Callers



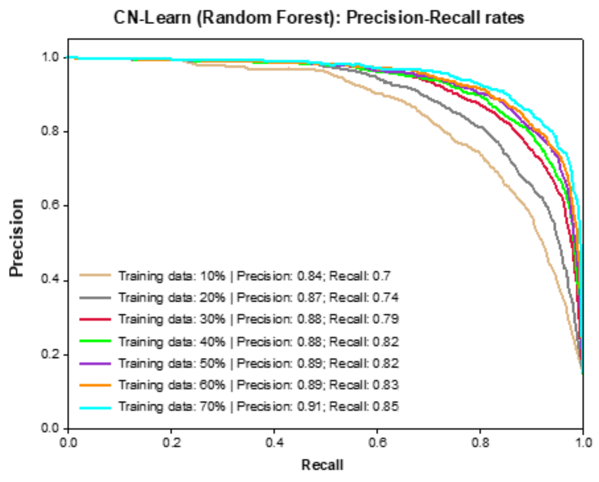
When performing these analyses, it is also interesting to examine the contribution of the various variables provided to the classifier. Interestingly, the quality control metrics passed to the various classifiers, Mappability and Read Depth Ratio, were among the more important features, at a combined 16%. The callers themselves, CANOES (17%), CLAMMS (8%), XHMM (4%), and CODEX (3%) have a combined 32% importance. Adding the concordance importance to this number yields a total 46%, which is less than what may have been expected. As Figure 3 implies, CLAMMS is a fairly important feature (8%), but is surprisingly overtaken by CANOES (17%) which had no singlet calls maintained by CNLearn in the sample run (Fig. 3). Unsurprisingly, CODEX and XHMM had relatively low importance compared to the other two callers.

Figure 4: Relative Importance of All Features Excluding CNV Frequency



Read depth ratio, another extremely important metric, was also extremely important when resolving read depth breakpoints. CNLearn has shown that with this relatively small number of features, extremely high precision and fair recall can be achieved. To avoid overfitting, a 70% training set was used, but CNLearn showed that it was fairly robust across different training set sizes.

Figure 5: Precision-Recall Across Training Data Proportions



The training data sizes here reflect percentages of the total population of 291 samples with microarray validation. As such, each 10% reflects roughly an additional 29 samples. While the 70% training curve is always superior to the other training proportions, the fact that the model maintained precision and recall values each within 20% of the values when the largest training data proportion was used. This result is extremely encouraging, as it may suggest that the validity of the results only suffers slightly when a limited amount of data is available. For this application, the Random Forest classifier appears to be the best choice due to the shape of its precision-recall curve. The other methods tested offer much poorer precision-recall rates at the 70% training data proportion. LR has 88% precision and 75% recall, while SVM has 82% and 75% precision and recall, respectively.

Discussion

In general, the high precision and recall rates found for CNLearn are encouraging and show that it can effectively recapitulate the microarray data results given the outputs of the four

CNV callers used. One of the more interesting results from the factor importance plot is the relative unimportance of some of the callers. CNLearn draws heavily from CANOES and CLAMMS at 17% and 8% respectively, but 50% importance is made up by factors independent of any individual caller. Namely, mappability, read depth ratio, chromsome number, GC content, target probe count, and size label.

This result suggests several directions for future improvements to this ensemble calling method. Firstly, it may suggest that CANOES and CLAMMS should be kept, but that other callers like EXCAVATOR, CoNIFER (Krumm, et al., 2012), or the recently published CODEX2 (Jiang, et al., 2018) could also be experimented with as additional sources of breakpoints. Further analysis could also be performed with eliminating one or more of the callers, especially XHMM and CODEX. Their extremely low importance (4% and 3%) and relatively low precision would suggest that these callers act more as a source of noise than they do a meaningful signal. This could crucially also cut down dramatically on the amount of time needed to process results, which could be a priority in situations where less computational power is available.

While CNLearn has proven to be fairly robust with various training set proportions, we have not proven that all of the individual CNV calling algorithms used within it would still provide usable results with small patient numbers. As mentioned in the Methods and Materials section, some of the callers that use ensemble methods to determine what normal and aberrant coverage depths are, and use this to determine the CNV status at each locus. If the total number of samples provided to these algorithms decrease, the output of the individual callers would suffer, likely causing CNLearn to similarly suffer. This issue can be avoided by always running larger numbers of samples through the individual callers, especially individuals with no medically relevant CNVs. This does increase the total computation time, but may be needed to ensure the best possible results.

While the individual calling algorithms do add somewhat to processing time, there are certain strategies that could be employed to reduce this. In previous runs, unpublished code experimented with parallelizing the process of calling the various CNVs, reducing wait time by running similar tasks simultaneously. By far one of the biggest time sinks is the long time required for read alignment because the original authors of each caller used different methods. XHMM uses GATK, CANOES uses bedtools, CODEX uses samtools, and CLAMMS relies on BWA for FASTA indexing. These steps are extremely time-consuming, and they all perform the same essential task.

As mentioned earlier, the use of microarray data as the validation set for CNV calls does expose some of the limitations of CNLearn. Microarrays are limited in that they can only interrogate small regions of the genome and their accuracy can be impacted by GC content. With the advent of long-read sequencing, it is possible that more accurate gold standards may become available. However, while this limitation currently represents an upper bound on the accuracy of CNLearn, this experiment does show that CNLearn can recreate calls made by an unrelated CNV calling method using only the output WES-based methods. While it has not been experimentally shown, this lends credence to the idea that CNLearn could easily adapt to receive inputs from higher quality sources of true CNV information. As with many machine learning programs, CNLearn is limited primarily by the quality of the training data available to it.

In terms of improving the performance of the individual CNV callers, several remedial steps could be taken. In the case of CODEX, the k-value used smoothen the data could be adjusted. Currently, the method used to generate the k-value is conservative, allowing for some false calls to ensure that it captures more true positives. Based on the fact that CODEX had such a large number of calls, it seems that a less conservative k-value selection method could be chosen so the number of false calls would be reduced. Moving forward, however, implementation of CODEX2, an updated CODEX program, should be implemented. CODEX is not maintained anymore, which could be an issue for those seeking to implement CODEX on newer paradigms (such as hg38) as they emerge. The other caller with low importance was XHMM, for which a clear solution does not present itself. XHMM made 649 calls, in the same neighborhood as CLAMMS, which made 596 calls. For this reason, reducing the amount of unsmoothed variation in read depth does not present itself as a good path to take moving forward. Additionally, it would be interesting to examine the ability of CNLearn to work across sequencing machine types. Namely, different models of sequencing machines or machines manufactures by different companies. The data used in this experiment all originated from SVIP, all using the Agilent SureSelect Human All Exon v2.0 capture kit and examining the same targets. As some of these factors change, the performance of CNLearn is expected to be affected, but the extent is unknown.

In general, CNLearn shows extremely good performance, with sensitivities and specificities that hold consistent across CNV size and rarity. While there are a host of potential optimizations that could be made to this program, it has proven to be a strong test case for ensemble-based machine learning methods for CNV calling. It currently shows a great deal of improvement over other CNV calling tools, and has extremely high precision, which would be critical if CNLearn or similar technologies would be used in a clinical setting.

BIBLIOGRAPHY

Asa, B.-H., Horn, D., Siegelmann, H., & Vapnik, V. (2001). Support Vector Clustering. *Journal of Machine Learning Research, 2*.

Backenroth, D., Homsy, J., Murillo, L., Glessner, J., Lin, E., Brueckner, M., . . . Shen, Y. (2014, 7). CANOES: detecting rare copy number variants from whole exome sequencing data. *Nucleic acids research, 42*(12), e97.

Bhardwaj, R., Nambiar, A., & Dutta, D. (2017). A Study of Machine Learning in Healthcare. *Proceedings - International Computer Software and Applications Conference. 2,* pp. 236-241. IEEE Computer Society.

Chen, Y., Liu, T., Yu, C., Chiang, T., & Hwang, C. (2013, 4 29). Effects of GC Bias in Next-Generation-Sequencing Data on De Novo Genome Assembly. *PLoS ONE, 8*(4).

Clancy, S. (2008). Copy Number Variation. *Nature Education, 1*(95), 1.

Fromer, M., & Purcell, S. (2014). Using XHMM software to detect copy number variation in whole-exome sequencing data. *Current Protocols in Human Genetics, 81*(SUPPL.81), 7.23.1.

Hastie, Trevor, Tibshirani, Robert, Friedman, J. (2009). *The Elements of Statistical Learning The Elements of Statistical LearningData Mining, Inference, and Prediction, Second Edition.*

Horning, N. (n.d.). *Random Forests : An algorithm for image classification and generation of continuous fields data sets.*

Jiang, Y., Oldridge, D., Diskin, S., & Zhang, N. (2015, 3 31). CODEX: a normalization and copy number variation detection method for whole exome sequencing. *Nucleic acids research, 43*(6), e39.

Jiang, Y., Wang, R., Urrutia, E., Anastopoulos, I., Nathanson, K., & Zhang, N. (2018, 11 26). CODEX2: Full-spectrum copy number variation detection by high-throughput DNA sequencing. *Genome Biology, 19*(1), 202.

Krumm, N., Sudmant, P., Ko, A., O'Roak, B., Malig, M., Coe, B., . . . Eichler, E. (2012, 8). Copy number variation detection and genotyping from exome sequence data. *Genome Research*, *22*(8), 1525-1532.

Mitchell, T. (2017). Tom Mitchell : Naive Bayes and Logistic Regression. *Machine Learning*, 1-17.

Packer, J., Maxwell, E., O'Dushlaine, C., Lopez, A., Dewey, F., Chernomorsky, R., . . . Reid, J. (2016). CLAMMS: A scalable algorithm for calling common and rare copy number variants from exome sequencing data. *Bioinformatics, 32*(1), 133-135.

Pearl, J. (2018, 1 11). Theoretical Impediments to Machine Learning With Seven Sparks from the Causal Revolution.

Segal, M. (2004). *Machine Learning Benchmarks and Random Forest Regression.*

Xia, X. (2010). *The Effect of Probe Length and GC% on Microarray Signal Intensity: Characterizing the Functional Relationship.*

Zhang, Z., Cheng, H., Hong, X., Di Narzo, A., Franzen, O., Peng, S., . . . Hao, K. (2019). EnsembleCNV: an ensemble machine learning algorithm to identify and genotype copy number variation using SNP array data. *Nucleic acids research, 47*(7), e39.

ACADEMIC VITA

Education

The Pennsylvania State University Schreyer Honors College, University Park, PA 2020
Bachelor of Science in Biomedical Engineering, Minor in Chemistry

Research and Work Experience

Intern and Employee at Girirajan Lab at Penn State 2015-2020

- Evaluating accuracy of various Copy Number Variation calling algorithms

Intramural Research Training Award Fellow at National Institutes of Health 2018 Summer

- Investigating methylation patterns in subtypes of ovarian cancer
- Ascertaining quality of mouse cancer as a model for human cancer

Summer Intern at Genentech 2019 Summer

- Evaluating clinical strength of novel eye health test
- Analyzing competitive strength of app within home monitoring market

Research

- Peer reviewed paper: Kumar, V., **Jayakar, G.,** Jensen, M., Kelkar, N., Girirajan, S. (2019). A machine-learning approach for accurate detection of copy-number variants from exome sequencing. *Genome Research*, 29: 1134-1143. Doi: 10.1101/gr.245928.118
- Poster: **Jayakar, G**., Giangreco, N., Petrykowska, H., Margolin, G., Gotea, V., & Elnitski, L. (2019). Differential Methylation and Ovarian Endometrioid Cancer: A Comparison Between GE Mice and Human. Poster presented at the National Institutes of Health Summer Poster Day, Bethesda, Maryland.
- Poster: **Jayakar, G.,** Haskova, Z., Willis, J. (2019). Review of Home Monitoring Devices in Ophthalmology. Poster presented at Summer Intern Poster Day, South San Francisco, California.

Skills

Programming

- Performing research using programming and scripting in Java, Python, R and awk
- Running genomic analysis programs, including GATK, CODEX, CLAMMS and other CNV tools

International Experience

- Intensive Korean course, Sogang University, with NSLI, U.S. Department of State, Summer 2014
- Spanish language courses on Cuban history and culture, Havana, Cuba, CIEE, Summer 2017.

Service, Leadership, and Recognition

Volunteer, Mount Nittany Medical Center 2010-2020

- Completed over 350 Hours of Volunteer Work & received scholarship

Academia 2016-2020

- Remote Area Medical clinic club (Secretary, 2019-2020)
- Penn State Quiz Bowl (Public Relations Chair, 2019-2020)
- Vollmer-Kleckner and Schreyer Honors College Scholarships (2016-2020)

Boy Scouts of America (Troop 31) 2005-2016

- Achieved the Rank of Eagle Scout