Classification of Genomic Variants from Whole Exome Sequencing with Machine Learning

Maria Simbirsky
CS 542
Summer 2018
Boston University
Boston, MA
msimbirs@bu.edu

Matthew Scott
CS 542
Summer 2018
Boston University
Boston, MA
mscott24@mgh.harvard.edu

Charlotte Chen
CS 542
Summer 2018
Boston University
Boston, MA
cc423@bu.edu

Abstract

Next Generation Sequencing (NGS) allows for the in depth investigation of cancer genomes. As whole genome sequencing transitions from research environment to clinical and industrial environment, one major challenge current sequencing tools are facing is the low accuracy in identifying true gene mutation in low coverage data. Using features extracted from established bioinformatics workflows, we implement a classifier to distinguish true mutations from false positives in a downsampled, low coverage dataset. Leveraging the advances that have been made in recent years in the field of machine learning, we implement two sets of classification algorithms to solve this problem. We implement both a soft-margin support vector machine (SVM) and a tuned random forest (RF) algorithm. Accuracy for both methods reached >95%. With these

1 Introduction

Due to advances in human genomic sequencing, the costs of having a patient's DNA sequenced and analyzed have dropped dramatically. Massive troves of human genomic data are being generated for both research and clinical purposes. In cancer, which is a "disease of the genome", mutations spontaneously arise in a patient's genome which lead to the proliferation of uncontrolled cell duplication, causing a tumor to form. The DNA of a tumor is typically mutated and is no longer identical to the DNA of the patient. Thus, a large body of research is dedicated to the identification of "somatic variants" - variants arising in the tumor linked to the cancer state. One method for studying the genomes of patients is Human Whole Genome Re-Sequencing (WGS), a popular technique which involves aligning raw patient sequencing reads against an existing human genome reference to identify differences, or variants, in the sample. However, a major challenge in identifying somatic variants arises from inconsistencies in sequencing. Many powerful somatic variant calling tools have been developed to identify cancer-linked mutations given a set of sequencing data from the tumor tissue and the patient's healthy tissue. However, these tools perform well only under optimal conditions, and suffer from significant loss of accuracy when using low quality or low depth sequencing data. Sequencing to very large depth, where a lot of raw data is generated for each section of the genome, is costly and impractical, and the depth across the genome is inconsistent due to the folded structure of DNA, making some parts of the genome inaccessible.

In this work, we explore whether machine learning can be used to improve the accuracy of low pass somatic variant identification by leveraging existing bioinformatics tools for variant identification and classifying, using machine learning, whether the identified variants are "true mutations" or "false positives", such as sequencing errors.

Preprint. Work in progress.

2 Pre-Processing

Data was obtained through the Texas Cancer Research Biobank (TCRB) Open-Access Database [1]. This data was made available in deidentified form from tumor and non-tumor tissue samples of cancer patients and consists of raw sequencing data produced on an Illumina HiSeq instrument. The full dataset for each patient was run using standard bioinformatic workflows. The reads were aligned to the GRCh38 human reference genome [4] using the BWA-MEM [9] algorithm. Only Single Nucleotide Variants - mutations where a single DNA nucleotide mutated to another - were examined in our dataset. Variants were called using three somatic mutation identification tools used commonly in the field: Strelka2 [6], SomaticSniper [7] and Sentieon TNsnv [3]. Any variants identified by all three tools using the full dataset were then classified as "true positive" variants. The dataset was downsampled randomly to include only 50% of the raw sequencing reads using the package seqtk [8]. The downsampled dataset was run through the same pipeline and the truth set was used to classify "true positive" variants for both the training and testing sets. Patient data from Case 002 was used as the training set and from Case 003 was used as the testing set, meaning our models were trained and tested on two different patients. Variants were determined to be equivalent if they were found to be in the same chromosome and position.

Features extracted from the data are listed in Table 1. The features used were chosen because these are statistics reported by the bioinformatics tools above in the variant report output. No additional features were calculated. A Python script was used to compare variants for the truth set and to extract the data from the variant report file format (VCF) to a Matlab compatible matrix.

Table 1: Features

Name	Description	Type	Value Range
Chromosome	Chromosome number of variant	string	varies
Position	Position of variant within chromosone	integer	varies
Transition	Whether the variant is a Transition (1) or a Transversion (0)	boolean	0 or 1
Tumor Allele	Nucleotide of tumor allele	string	A, C, T, or G
Normal Allele	Nucleotide of normal allele	string	A, C, T, or G
Strelka	If Strelka2 identified this variant	boolean	0 or 1
Sentieon	If Sentieon identified this variant	boolean	0 or 1
SomaticSniper	If SomaticSniper identified this variant	boolean	0 or 1
Genotype	The tumor/normal genotype of this variant	string	varies
Somatic	Whether this mutation is somatic	boolean	0 or 1
Depth	Number of reads covering or bridging the variant site	numeric	varies
Map Quality	Average read alignment score to the reference	numeric	varies
Map Quality 0	Number of reads with 0 mapping quality	numeric	varies
Base Quality	Per-nucleotide accuracy score in the raw reads	numeric	varies
Variant Quality	Normalized accuracy score of the reported genotype	numeric	varies
Somatic Quality	Normalized accuracy score of the somatic allele	numeric	varies

3 Models

Two general sets of algorithms were used to solve our classification problem: support vector machines and random forests/decision trees. A support vector machine was implemented using a soft-margin classification schema with both a linear and non-linear kernel. Moreover, a tuned random forest and decision tree algorithms were used. As a reference for performance, standard logistic regression was assessed for comparison.

3.1 Support Vector Machines

Support vector machines (SVM) are a supervised machine learning algorithm that can be used for both classification or regression. With our task, we reusing it for a classification problem. In this algorithm, we present points in $\mathbb{R}^n$ and perform classification by finding the maximum-margin hyperplane that differentiate the two given classes [2]. The standard linear classification problem for SVM is

$$y(\overrightarrow{x}) = \overrightarrow{w}^T \overrightarrow{\phi}(\overrightarrow{x}) + b \quad Bishop\ (7.1)$$

with $\overrightarrow{x}$ as our features, $\overrightarrow{\phi}$ as the transformation to our feature space, b as the bias, and y as our classification schema. Furthermore, since our labels are only of two classes and this is a binary problem, $\overrightarrow{x}$ are classified according to the sign of $y(\overrightarrow{x})$. With t_n denoting true class labels,

$$t_n(\overrightarrow{w}^T \overrightarrow{\phi}(\overrightarrow{x_n}) + b) \geq 1 \quad Bishop(7.5)$$

However, this is only the case if we are classifying our data with a strict requirement of the support vectors. This can lead to overfitting our data significantly. Albeit, there does exist a method that can ease the restrictiveness of our margins. To do this, we introduce what is called a 'slack' variable to make this a 'soft-margin' SVM problem. This hyperparameter adds the possibility that certain points not falling explicitly on our support vectors can now be included. We see this by

$$t_n(\overrightarrow{w}^T \overrightarrow{\phi}(\overrightarrow{x_n}) + b) \geq 1 - \xi_n,\ \xi_n \geq 0 \quad Bishop(7.20)$$

This new constraint allows a margin that is less than 1, and it contains a penalty $C \sum_n^N \xi_n$ for any data points that are not within the margin on the correct side of the 'street' so to say. We thus can state our preference for margins that classify the training data correctly, but in addition we can soften the constraints to allow for non-separable data with a penalty proportional to the amount the example is misclassified. The parameter C moderates the relative weighting between the aim of making the margin small and ensuring that most examples have margins that are at least 1. Therefore, we would like to minimize the sum total of the penalties ξ_n over all n (the upper bound on the training classification errors). Finally, the new, L1-regularized, well-posed optimization problem becomes the objective function

$$min\ \frac{1}{2} \parallel \overrightarrow{w} \parallel^2 + C \sum_n^N \xi_n \quad Bishop(7.21)$$

$$t_n(\overrightarrow{w}^T \overrightarrow{\phi}(\overrightarrow{x_n}) + b) \geq 1 - \xi_n,\ \xi_n \geq 0 \quad Bishop(7.20)$$

In addition to using a slack variable for our SVM, we will look at the effects of both a linear and non-linear kernel on testing accuracy. For this we use a simple linear kernel as well as a Gaussian kernel, also known as a Radial Basis Function (RBF). Broadly speaking, kernels allow our feature space to be transformed to a new dimensional space that will allow our margins to classify groups that were not separable before. The two kernels take the form of

$$k(\overrightarrow{x_n}, \overrightarrow{x_m}) = c\overrightarrow{x_n}^T \overrightarrow{x_m} \quad Bishop(6.13)$$

$$k(\overrightarrow{x_n}, \overrightarrow{x_m}) = exp(- \parallel \overrightarrow{x_n} - \overrightarrow{x_m} \parallel^2 / 2\sigma) \quad Bishop(6.23)$$

We first optimized our objective function to find hyperparameters for our SVM using both a linear and Gaussian kernel. Plots of this optimization are shown in Figure 1 and Figure 2. Here, we see that the radial basis function had a much more distinct global minimum than the linear kernel. Thus, there was less variability in our choice of hyperparameters for the Gaussian kernel in comparison to the linear kernel. With the dynamics of these hyperparameters understood, we can then train our entire training set. After, testing accuracy is assessed for each method. All SVM implementations are completed in MATLAB using the Statistics and Machine Learning Toolbox.

3.2 Decision Tree & Random Forest

Decision tree [5] is another robust supervised machine learning algorithm, based on the concept of entropy and information gain. We first fitted a decision tree with the entire data. Model results shows that the most influential attribute to determine how to classify true mutation vs a false positive is the somatic attribute.

To overcome the issue of overfitting in a decision tree, we use Random Forest (RF) [10] and tuned one of the parameters to improve prediction accuracy. RF is a versatile ensemble learning method for

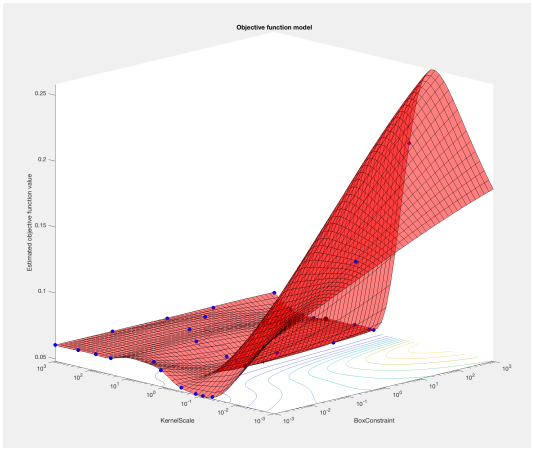

Figure 1: Linear kernel optimization with slack constraint (right), kernel parameter (left), and objective function evaluation (vertical)

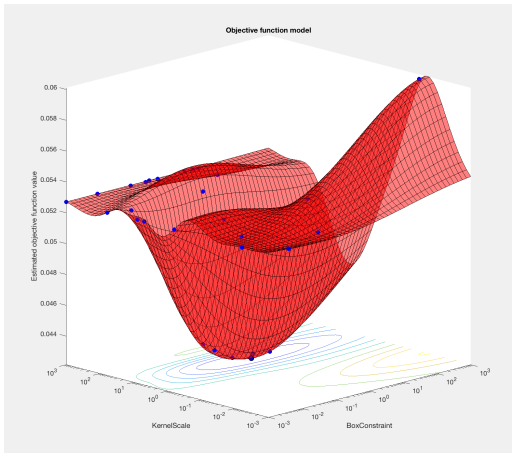

Figure 2: RBF kernel optimization with slack constraint (right), kernel parameter (left), and objective function evaluation (vertical)

classification, regression, and other response types. The method operates by constructing numerous decision trees and averaging or "bagging" across these trees, selecting features (or variables) that appear towards the upper branches of each fitted tree and therefore are predictive of response. Note only is it used for prediction, but also to assess feature importance, which indicates how much including a particular variable improves the prediction.

Tuning the parameters in random forest will help increase the predictive power of the model or make it easier to train the model. In our case, we tuned the maximum number of features random forest used to create individual trees. Since the default number is 10, which results in an error rate of 4.14%, we tested model accuracy from 10 to 20, in which 14 outperformed the rest by a small margin resulting in an error rate of 4%.

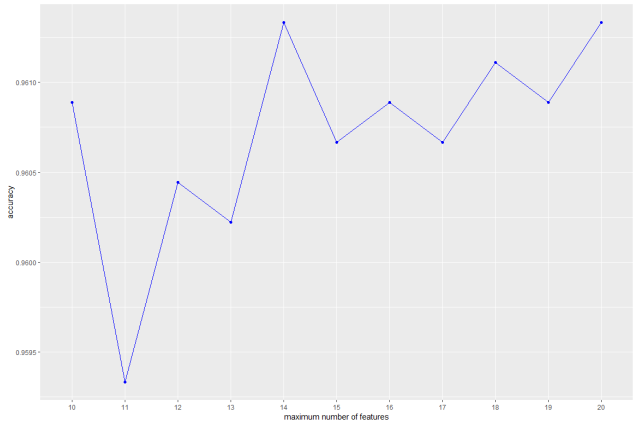

Figure 3: RF parameter tuning (horizontal) with accuracy (vertical)

All training and tuning are in R. Main libraries used are randomForest,rpart, caret, dplyr and ggplot2 as well.

4 Results

We implemented three categories of algorithms with five results total and they are shown in Table 2. above. SVM with a Gaussian kernel was found to be the most accurate due to its accuracy of 96.375%. However, SVM with a linear kernel was a close second with 96.332% accuracy. Following

Table 2: Accuracy measures

Method	Accuracy
SVM with Linear Kernel	96.332%
SVM with Radial Basis Kernel	96.375%
Random Forest	96.133%
Decision Tree	95.869%
Logistic Regression	94.601%

this, random forest was third with an accuracy of 96.133%. After random forest was decision tree with 95.869%. With these results from standard Machine Learning algorithms, we can compare these to more conventional general linear models. Specifically, logistic regression was shown to have an accuracy of 94.601%. When looking at the range between SVM with a Gaussian kernel and logistic regression (the highest and lowest accuracy), we see that there is only a difference of 1.73%.

5 Discussion and Conclusion

With 96.375% accuracy from SVM with a Gaussian kernel and other high performing techniques listed in Table 2, we seem to be able to utilize machine learning algorithms to identify true gene variation with machine learning algorithms. Below in depth we present the results from each algorithm.

5.1 Kernel Variation with SVM

One primary computational item of interest in our investigation was the effect of a linear and non-linear kernel. We implemented both a linear and RBF kernel to solve this. In Figure 4. we see a comparison of each kernel's accuracy over ten iterations. Qualitatively, we see that the RBF kernel is generally above the linear kernel in accuracy. Quantitatively, the average accuracy for the RBF kernel is 96.19% and 96.06% for the linear case. Thus we have an average difference of 0.13% over these ten iterations.

Though we do see a distinct difference between the two, both returned an accuracy in the domain of 96.3%. The next logical question to ask then is if this has any implication on our biological problem. Our testing data consisted of 354,703 mutation classifications. With a 0.13% average difference in these iterative comparisons, about 461 mutations could be recovered if the non-linear kernel was used instead. Though factors such as time and computational power may need to be accounted for, there does not seem to be a reason to use a linear kernel otherwise.

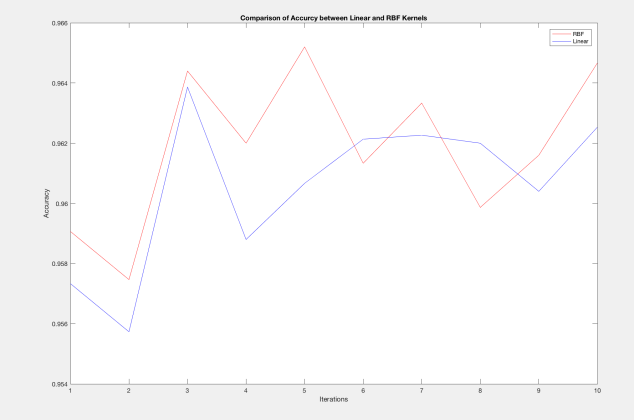

Figure 4: Comparison of kernel accuracy's (vertical) over multiple iterations (horizontal). The RBF kernel is in red and the linear kernel is in blue. Average accuracy for the RBF kernel is 96.19% and 96.06% for the linear case. Each iteration was taken over a sub-sample of N=50,000 with replacement.

5.2 Decision Tree and Random Forest

Random forest (96.133%) slightly outperformed decision tree (95.869%) but by only a very small margin. Additionally, the decision tree (95.869%) results shows that the most important feature to predict if a mutation is true is the somatic nature, which is the third most important feature from random forest. Overall, true gene mutations exist mostly on those mutations whose somatic value is 1, sentieon value is 1, and depth smaller than 3.5.

As a collection of trees, random forest has the ability to limit over fitting without substantially increasing error due to bias so the prediction is better. As shown in the Figure 6, the most important features emerged from random forest is the sentieon field, which is the second most important feature from decision tree as well. Overall, these two model have similar results regarding both prediction accuracy and feature importance ranking.

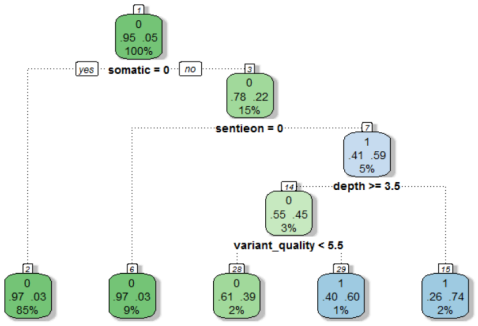

Figure 5: Graphical representation of decision three results

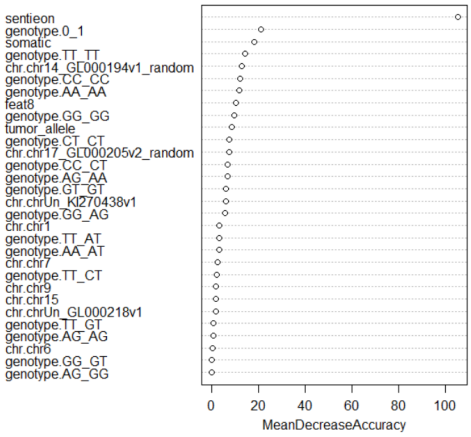

Figure 6: Results of random forest with features (vertical) and mean decreased accuracy (horizontal)

5.3 Comparison to Conventional General Linear Models

As stated in Table 2. logistic regression achieved 94.601% accuracy on the testing data set. With only a difference of 1.71% from the most accurate SVM, this brings the question of computation expense and time into question. Similar to our discussion of kernel choices, a 1.71% difference from a 353,703 testing data set can lead to the classification of over 6,4832 mutations. However, the computational expense for this regression may be important in comparison to that of more robust machine learning methods. Though not ideal, in situations that may require an accessible, computationally inexpensive solution this is certainly a viable alternative.

5.4 Limitations and Future Improvements

All models, including logistic regression, achieved prediction accuracy greater than 94.000%. With these results, we wonder if such a high accuracy rate may be caused by data bias. As stated before, the gold standard truth set was generated by consensus from the three tools and cannot be biologically validated without a real truth set. In addition, due to the nature of the processing, many of the features extracted are highly correlated to each other. For example, a somatic variant being called by a given tool is highly dependent on the read depth and average mapping score, as those are the features used by the tool itself. Even if this was the case though, the classification rate would be still correct. However, it would be less so the responsibility of the classification algorithm and more so the effect of the processing software itself.

Finally, this algorithm can be used to detect "true positives" from "false positives" but cannot extract hidden features from the data, such as variants that were not called by the three tools used. A truly powerful approach to processing low coverage data would also be able to detect true and false negatives.

We have provided only an introduction to the potential machine learning can bring to genomics data. Moving forward, the goal is to cover the last 3% to 4% of sequencing accuracy on low coverage reads. We believe it is entirely possible to get higher accuracy using more robust algorithms as shown though the accuracy of SVM and Random Forest. Aspiring to leverage the advancements being made in the field of machine learning moving forward, we hope that implementing more advanced methodologies can produce more accurate results.

Acknowledgements

We thank the Texas Cancer Research Biobank and the Baylor College of Medicine for making the raw TCRB data publicly available.

Credits

Pre-processing and data loading code was implemented by M. Simbirsky. SVM implementation and parameter optimization was done by M. Scott. Random Forest and Linear Regression analysis was performed by C. Chen. The manuscript was written by all contributing authors.

References

[1] T. C. R. B. Tcrb open access data, 2014. https://dnanexus.github.io/tcrb-data/[Accessed: Aug 2018].

[2] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

[3] D. N. Freed, R. Aldana, J. A. Weber, and J. S. Edwards. The sentieon genomics tools - a fast and accurate solution to variant calling from next-generation sequence data. *bioRxiv*, 2017. doi: 10.1101/115717. URL https://www.biorxiv.org/content/early/2017/05/12/115717.

[4] Y. Guo, Y. Dai, H. Yu, S. Zhao, D. C.Samuels, and Y. Shyr. Improvements and impacts of grch38 human reference on high throughput sequencing data analysis. *Genomics*, 109:83–90, 2017. doi: https://doi.org/10.1016/j.ygeno.2017.01.005.

[5] S. Hartshorn. *Machine Learning With Random Forests And Decision Trees: A Visual Guide For Beginners)*. Independently published, 2016. ISBN 978-1549893759.

[6] S. Kim, K. Scheffler, A. L. Halpern, M. A. Bekritsky, E. Noh, M. Källberg, X. Chen, D. Beyter, P. Krusche, and C. T. Saunders. Strelka2: Fast and accurate variant calling for clinical sequencing applications. *bioRxiv*, 2017. doi: 10.1101/192872. URL https://www.biorxiv.org/content/early/2017/09/23/192872.

[7] D. E. Larson, C. C. Harris, K. Chen, D. C. Koboldt, T. E. Abbott, D. J. Dooling, T. J. Ley, E. R. Mardis, R. K. Wilson, and L. Ding. Somaticsniper: identification of somatic point mutations in whole genome sequencing data. *Bioinformatics*, 28(3):311–317, 2012. doi: 10.1093/bioinformatics/btr665. URL http://dx.doi.org/10.1093/bioinformatics/btr665.

[8] H. Li. Seqtk: a fast and lightweight tool for processing fasta or fastq sequences. https://github.com/lh3/seqtk, year = "2018".

[9] H. Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. 1303, 03 2013.

[10] S. M. Ram M, Najafi A. Classification and biomarker genes selection for cancer gene expression data using random forest. *Iranian Journal of Pathology*, 12(4):339–347, 2017. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5844678/.