Genetic Variant Call Filtering

Nico Chaves

All relevant code for the project can be found at:
https://github.com/nmchaves/variant-call-filter

I. Background

When an individual's genome differs from a reference genome at some location, we say that the individual has a "genetic variant" at that location. When a sequencing pipeline finds evidence for a variant, it outputs a "variant call" describing the variant. Obtaining an accurate set of variant calls for an individual is crucial to clinical genomics. However, in practice, variant call sets tend to contain many false positives. In other words, the genome of interest often does not contain many of the called variants. For example, in the dataset I used for this project, approximately 10-15% of the variant calls are false positives.

There are 2 approaches to improve the quality of a variant call set. The first approach involves improving the upstream processes. For example, one may sequence the genome at higher coverage, integrate multiple sequencing methods to improve robustness, and integrate the individual's pedigree information (i.e. sequence the individual's relatives and take into account inheritance constraints) [1]. However, this would be prohibitively expensive to perform for every genome we would like to sequence. A more practical approach toward improving the quality of the variant call set is to develop a binary classifier that filters out the false positives while retaining as many of the true positives as possible. I pursue this second approach in this project.

To build such a classifier, I use features associated with the variant calls. Features are also referred to as "variant annotations" in this context. The features are generated by upstream processes, and they include: depth, quality score, strand bias, and several others. Briefly, depth is the number of reads supporting the variant call. A quality score represents the probability that a base was called correctly. The quality score for a given variant increases as the number of reads supporting that variant increases. As a result, it is useful to consider quality score/depth as a feature to capture the "average" quality of the reads supporting a given variant. Strand bias in sequencing refers to when one strand of DNA is favored over the other; a high strand bias may be indicative of a sequencing error. One can read about these and other features in the "Annotation Modules" section on the Genome Analysis Toolkit website [2].

II. Related Work

A. Genome Analysis Toolkit

The Genome Analysis Toolkit (GATK) [3] is a set of software tools for processing genomics data. The GATK includes a variant filtering tool called the Variant Quality Score Recalibrator (VQSR). The name reflects the fact that VQSR assigns a recalibrated quality score to each variant call. This new quality score is used to filter out low quality variant calls.

VQSR uses the features associated with the variant calls (including the original quality score) to train a Gaussian mixture model (GMM). VQSR only trains the GMM on variants which are known to exist in humans – for example, those that are found in the HapMap dataset [4]. In general, called variants which are already known to exist are more likely to be true positives. Thus, the GMM attempts to capture the various ways in which true positive variants cluster in the feature space. Let Q' denote the recalibrated quality score of a given variant. Q' is the maximum probability density of the variant, where the max is taken over all of the Gaussians in the GMM.

After building the GMM, VQSR filters out the variants with low Q', i.e. those with low probability density under *all* of the Gaussians. In particular, the user specifies a certain sensitivity threshold, such as $T = 99\%$. Then VQSR chooses a threshold Q_{99} such that 99% of the variants that it trained on satisfy $Q' \geq Q_{99}$. Clearly, if the user specifies a lower sensitivity threshold T, then Q_T will increase and VQSR will have fewer true positives and fewer false positives.

III. Datasets

I used data contained in the GATK resource bundle [5]. In particular, I used variant calls from chromosomes 11 and 20 of individual NA12878. Chromosome 20 contains approximately 63 million base pairs, or approximately 2% of the total number of base pairs in the human genome. Chromosome 11 is approximately twice as large, containing approximately 135 million base pairs. This dataset was generated using Illumina HiSeq technology at relatively low coverage (roughly 15x). Chromosome 20 (11) contains 86,843 (187,242) variant calls, 83.4% (93.5%) of which are true positives according to the gold standard dataset.

A. Gold standard Dataset

The gold standard dataset was generated using multiple sequencing techniques. The chromosome 20 data consists of 89,426 high confidence variant calls from NA12878 [6]. By definition, we assume that these are all real variants. Furthermore, we assume that any true positive variant found in the non-gold-standard dataset can also be found in the gold standard dataset. In other words, we assume that the gold standard dataset has not failed to identify any true variants that the non-gold-standard dataset has identified.

Note that the gold standard dataset contains more variants than the non-gold-standard dataset. This means that even if we filter the non-gold-standard dataset perfectly, we will still fail to identify some of the variants in the individual. However, the goal of this project is simply to filter out the false positive variant calls, not to discover false negatives.

B. Preprocessing

The raw variant call dataset is in VCF format, which is difficult to process, so I converted it to a tab-separted file using the GATK's VariantsToTable tool. When using this tool, I split multi-allelic variant calls into separate variant calls. For example, a variant call with bases A and T would be split into one variant call with base A and another variant call with base T. Multi-allelic variant calls are difficult to process without splitting, because some of their features are tuples. Moreover, the chromosome 20 dataset only has approximately 20 multi-allelic variant calls, so splitting them will not have much effect on the overall results.

I then removed features with nonnumeric values. Some features do not have values for all variants (which results in "Not a number" values), and other features are represented as strings. Note: in the future, I intend to embed the string features as categorical variables. After this step, the following features remained (the corresponding VCF annotation symbols are given in parentheses):

1) quality score (QUAL)
2) allele count (AC)
3) allele frequency (AF)
4) depth (DP)
5) strand bias as measured by Fisher score (FS)
6) maximum likelihood expectation for the allele counts (MLEAC)
7) maximum likelihood expectation for the allele frequency (MLEAF)
8) root mean square mapping quality (MQ)
9) quality score normalized by number of reads supporting the variant (QD)
10) strand bias as measured by symmetric odds ratio (SOR)

Note that the QUAL feature here is the original quality score, not the score obtained from using VQSR.

I also used 2 binary features: (1) whether the variant is in the dbSNP dataset and (2) whether the variant was treated as a positive training site by VQSR (for example, this feature would be "1" if the variant is found in HapMap). These two features capture whether or not the variant is "common". Note that 97.6% of the variants in the dataset are found in dbSNP and 75.4% are treated as positive training sites by VQSR.

As a final preprocessing step, I scaled the features (except for the binary features) to have zero mean and unit variance, which some algorithms require.

C. t-SNE Embedding

After preprocessing the data, I generated t-SNE embeddings of the chromosome 20 variant call dataset, as shown in Figure 1. Note that the axes in the embeddings are unitless. The data points are colored according to whether they are considered true positives by the gold standard dataset. To reduce computation time, I performed each embedding on a randomly selected subset of the variants. Figure 1(a) was generated from 5,000 randomly selected variants (4,217 of which are true positives), while Figure 1(b) was generated using 500 randomly selected variants (398 of which are true positives).

We observe that some of the false positives cluster together in the embeddings. This is especially apparent in Figure 1(b). This suggests that we should be able to build a binary classifier that filters out some of the false positives. However, there are still a large number of false positives dispersed widely around both embeddings. This suggests that the data is not separable, and we should not expect to attain perfect classification.

IV. Experiments

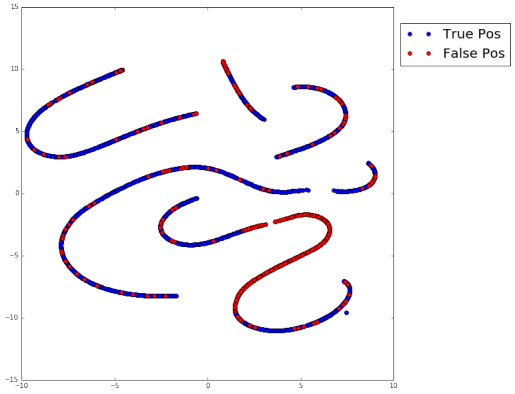
A. VQSR

To train VQSR, I used the 1000 Genomes project for the reference genome [7]. I also used the HapMap, dbSNP, and OMNI 2.5 datasets as described in the GATK best practices for training VQSR to filter SNP calls [8].

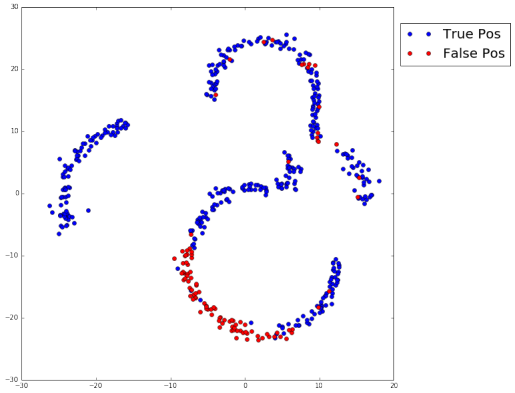
As an example, VQSR chose 75.4% of the variants (or 65,506 variants out of the total 86,843 variants) from the NA12878 chromosome 20 dataset for training the GMM.

Note that VQSR does not use a supervised learning approach, so it does not have a test set per se. However, I still evaluated it on the same test set as the supervised learning algorithms so that the methods can be compared on the same set of variants.

Figure 1. t-SNE Embeddings



(a) 5,000 randomly selected variants. Perplexity=200.



(b) 500 randomly selected variants. Perplexity=30.

B. Supervised Learning (Single Chromosome)

I used supervised learning algorithms to build a binary classifier that predicts whether a variant call is a true positive or not based on the features listed in section 3c. In particular, I used logistic regression, SVM (with Gaussian kernel), and random forest models. I used L2 regularization for both logistic regression and SVM.

I trained these supervised learning algorithms on a randomly selected 70% training set (60,790 variants) from chromosome 20 of individual NA12878. I used 5-fold cross validation and grid search to select the best hyperparameters for each model. I used the F1 score as the metric for determining the best model during cross validation. I then evaluated the performance of the trained models on the held-out test set, i.e. the remaining 30% (26,053 variants) of the variant calls.

For random forest, I used 20 trees. I tried using up to 50 trees, but the validation error remained similar, while the execution time increased.

C. Supervised Learning (Multiple Chromosomes)

One may argue that training and testing a model on different variants from the same chromosome may overestimate performance due to correlations in the data. As a result, I also trained supervised models on chromosome 11 of NA12878 and used the models to filter variants from chromosome 20. In other words, I used chromosome 11 as the training set and chromosome 20 as the test set. Otherwise, I used a similar methodology to train the models as described in the previous section.

Even though the chromosome 11 and chromosome 20 datasets come from the same individual and sequencing technology, they still have some important differences. For example, the chromosome 20 dataset has ~83% sensitivity, while the chromosome 11 dataset has ~93% sensitivity. In other words, ~83% of the variant calls in chromsome 20 are also found in the gold standard dataset, i.e. ~83% of the variant calls are true positives. Since I trained on chromosome 11, one should expect the models to be biased toward the positive class and produce many false positives on the chromosome 20 test set. This difference between the 2 chromosome datasets means that the multiple chromosome experiments will be a more accurate representation of how well the supervised models generalize on new data.

D. Quasi-Supervised Learning

In reality, the number of gold standard datasets is limited. Ideally, one would be able to train a model which does not rely on a gold standard dataset. VQSR is an example of such an approach. As discussed already, VQSR determines which variants to train on by examining their features, such as membership in dbSNP. I ran experiments using this as a "quasi" label. If VQSR treated some variant as a positive (negative) training site, then I gave the variant a positive (negative) label at training time. I then trained supervised models as in the previous sections to apply filtering to the same variants which were trained on. I then evaluated the performance of this approach by using the labels in the gold standard dataset.

One may suggest using dbSNP membership itself as the quasi label. Unfortunately, ~98% of the variants called in the chromosome 11 and 20 datasets are found in dbSNP. As a result, any model trained using dbSNP models would be too skewed toward positive predictions to be useful.

E. Execution Time

The execution times required by each approach are shown in Table I. The total execution time for VQSR consists of training the GMM and applying the filter. The total execution time of the supervised algorithms includes the time to select the model parameters (using 3-fold cross validation for SVM and 5-fold cross validation for logistic regression and random forest), the time

Table I
EXECUTION TIME OF EACH ALGORITHM

	Total Execution Time (seconds)
VQSR	767.7
Logistic Reg	29.9
SVM	449.1
Rand Forest	117.9

Note: Times apply to the single chromosome case (i.e. training and testing on chromosome 20).

to retrain the model on the training set, and the time to evaluate the model on the test set. Note that one could reduce the execution time of the supervised algorithms by restricting the hyperparameter searches to a narrower range. However, restricting the model in this way may also reduce the performance.

While the times in Table I seem relatively short, one should keep in mind that the chromosome 20 training set is very small. Chromosome 20 accounts for ~2% of the total number of bases in the human genome. Therefore, one would need approximately 50x the above times to train on an entire human genome.

Finally, note that VQSR must be trained on each new dataset it encounters, whereas the supervised learning algorithms only need to be trained once for a given sequencing technology and processing pipeline (at least in principle). In the future, I plan to evaluate whether a trained model can actually be applied to a different dataset (e.g. a variant call set from a different individual, but using the same sequencing technology and processing pipeline).

V. RESULTS

A. Single Chromosome

Table II summarizes the performance of each approach in the single chromosome experiments. VQSR (99%) refers to VQSR with a sensitivity threshold of 99%, and similarly for VQSR (99.9%). In other words, VQSR (99%) means that VQSR sets its threshold Q_{99} such that 99% of the variants it trained on are classified as true positives. This does *not* necessarily mean that VQSR (99%) will achieve 99% sensitivity overall. The 99% and 99.9% thresholds are automatically generated when following the GATK's best practices.

All of the models are strongly biased toward false positives. In other words, the models retain far more variants than they filter. As shown in Figure 1, some of the false positives are spread among relatively dense regions of true positives. This suggests that even a very sophisticated model may still not be able to avoid false positives.

The three supervised learning algorithms performed very similarly, and they generally all performed better than VQSR. In particular, the supervised learning approaches led to higher F1 scores. SVM might have performed better if I searched over a larger parameter space during the training phase. However, SVM already took the longest to train, as shown in Table I. As a result, I chose not to increase its hyperparameter search space.

Table II
SINGLE CHROMOSOME PERFORMANCE COMPARISON

	F1	Precision	Recall
VQSR (99%)	0.925	**0.981**	0.924
VQSR (99.9%)	0.924	0.894	0.970
Logistic Reg	0.975	0.970	0.980
SVM	0.979	0.973	**0.986**
Rand Forest	**0.980**	0.976	0.983

The supervised models were trained on 70% of the chromosome 20 variant calls. The metrics above were computed on the held-out test set (i.e. the remaining 30% of chromosome 20).

Table III
MULTIPLE CHROMOSOME PERFORMANCE COMPARISON

	F1	Precision	Recall
VQSR (99%)	0.925	**0.981**	0.924
VQSR (99.9%)	0.924	0.894	0.970
Logistic Reg	**0.969**	0.951	**0.988**
SVM	0.726	0.858	0.630
Rand Forest	0.954	0.971	0.938

The supervised models were trained on chromosome 11 variant calls. The metrics above were computed on the held-out test set (i.e. chromosome 20).

B. Multiple Chromosomes

Table III summarizes the performance of each approach in the multiple chromosome experiments.

The logistic regression and random forest models still achieved a better F1 score than VQSR. However, SVM generalized very poorly, especially in terms of recall. Given that SVM also takes the longest to execute, this suggests that logistic regression and random forest should be prefered for this task.

C. Quasi-Supervised Learning

Table IV summarizes the performance of each approach in the quasi-supervised learning experiments.

Table IV
QUASI-SUPERVISED PERFORMANCE COMPARISON

	F1	Precision	Recall
VQSR (99%)	0.925	0.981	0.924
VQSR (99.9%)	0.924	0.894	**0.970**
Logistic Reg	**0.947**	0.966	0.929
Rand Forest	0.936	**0.991**	0.886

Logistic regression and random forest were trained using labels from VQSR on chromosomes 11 and 20. They were then evaluated directly on chromosome 20 using the gold standard labels.

Again, the logistic regression and random forest models achieved a better F1 score than VQSR. I was not able to develop a reasonable SVM model in this case, so I ommited it in the results. I plan to resolve this issue in my future work.

VI. Conclusions and Future Work

This project has shown that supervised learning may be preferable over VQSR for improving the quality of genetic variant call sets. For example, random forest and logistic regression achieved better F1 scores than VQSR in all 3 experiments, and they took significantly less time to execute. The multi-chromosome and quasi-supervised results are particularly encouraging, as they are a more realistic representation of performance. More sophisticated supervised learning approaches, such as deep learning, may attain even better performance. I intend to test this in the future.

I also plan to continue evaluating the current models on different datasets, such as variant calls from the remaining chromosomes of NA12878 and from different individuals (but using the same sequencing technology and processing steps).

As mentioned previously, I removed some nonnumeric features, which I plan to embed as categorical features in the future. As the number of features grows, feature selection will become more important. I intend to experiment with principal component analysis and mutual information feature selection. I also plan to try using these derived features as input to VQSR. This set of features may outperform the set of features recommended in the VQSR best practices.

VII. Acknowledgements

I'd like to thank Professor Tsachy Weissman, Professor Idoia Ochoa and Mikel Hernaez for their feedback and guidance.

References

[1] Eberle, Michael A., et al. "A reference dataset of 5.4 million human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree." bioRxiv (2016): 055541.
[2] VCF Feature Descriptions https://software.broadinstitute.org/gatk/gatkdocs/org_broadinstitute_gatk_tools_walkers_annotator_VariantAnnotator.php.
[3] McKenna, Aaron, et al. "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data." Genome research 20.9 (2010): 1297-1303.
[4] Gibbs, Richard A., et al. "The international HapMap project." Nature 426.6968 (2003): 789-796.
[5] GATK Bundle https://software.broadinstitute.org/gatk/download/bundle
[6] Zook, Justin M., et al. "Integrating human sequence data sets provides a resource of benchmark SNP and indel genotype calls." (2014).
[7] 1000 Genomes Project Consortium. "A global reference for human genetic variation." Nature 526.7571 (2015): 68-74.
[8] VQSR Best Practices https://software.broadinstitute.org/gatk/gatkdocs/org_broadinstitute_gatk_tools_walkers_variantrecalibration_VariantRecalibrator.php