A Generative Model for Distributed Read-Error Correction

Frank Austin Nothaft, Anthony D. Joseph, and David A. Patterson

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720, USA
{fnothaft, adj, pattrsn}@berkeley.edu

Abstract. Read error correction is a key step in many genome assembly protocols, but is computationally expensive. Recent papers have proposed several strategies for approximating steps in the error correction process. In this paper, we derive a rigorous generative model for identifying and correcting errors in short reads. We implement our algorithm using the Apache Spark distributed computing platform. Our implementation demonstrates a performant and exact approach to read error correction that can scale efficiently to mammalian genomes.

Keywords: genome assembly, read error correction, distributed computing

1 Introduction

Read error correction is a key step in most genome assembly protocols, but is computationally expensive [4]. Recent work has explored mechanisms for performing error correction with probabilistic data structures which consume less memory [9, 8, 5, 3]. However, we assert that these techniques are unnecessary with the rise of distributed, in-memory computing frameworks that scale easily on commodity hardware [10].

In this paper, we present a distributed algorithm for performing read error correction, which is built using the ADAM libraries for distributed genomics [6]. Our algorithm trains a model for read errors using the abundance spectrum of quality score weighted k-mers (known as q-mers [4]). Given these probabilities, we estimate transition probabilities across classes which contain correlated errors. Finally, per read, we perform error correction via a coordinate ascent process. As our algorithms are implemented on top of the Apache Spark in-memory MapReduce system [10], we are able to achieve both scale up and scale out performance. Our software is open source under the Apache 2 license.

We make the following contributions:

1. We implement a MapReduce based, in-memory k/q-mer counting engine.
2. We demonstrate a probabilistically rigorous, generative model for read error correction.
3. Our model is implemented in a scalable fashion on an efficient, MapReduce engine.

2 Generative Model for Error Correction

2.1 Erroneous k-mer Detection

To detect erroneous k-mers, we use the q-mer construct which was introduced by Kelley et al. [4][1] We apply an unsupervised clustering model to learn the classifications of q-mers. For a sample with ploidy of m, we model q-mers as being drawn from a selection of $m+1$ Gamma distributions. $m-1$ of these distributions model "true" q-mers which are drawn from heterozygous sites, one distribution models "true" q-mers drawn from homozygous sites, and one distribution models erroneous q-mers. While prior work by Kelley et al. [4] has used the Normal distribution to model the counts of "true" q-mers, it is preferable to use the Gamma distribution, whose support is limited to $[0, \infty)$.

We describe the process we use for counting q-mers in §3.1. Once we have counted q-mers, we then fit our mixture model using the expectation-maximization (EM) algorithm introduced by Almhana et al. [1] We considered the optimized algorithm for fitting mixtures of Gammas that was proposed by Schwander and Nielsen [7], but chose the Almhana EM algorithm because the Schwander k-MLE algorithm is slower for mixtures that contain fewer than eight components. We describe the implementation of this algorithm in §3.2.

Once the mixture model has been fit, each k-mer can be assigned an error probability. This probability is given in equation 1, and is defined by the softmax likelihood that a k-mer is drawn from the error distribution, given the q weight of the k-mer.

$$P(k \text{ is erroneous}) = \frac{\mathcal{L}(s = 0|k)}{\sum_{i=0}^{m+1} \mathcal{L}(s = i|k)} \tag{1}$$

Once the error probabilities have been calculated for all k-mers, we then filter the set of k-mers to obtain the set of trusted k-mers. We apply the following filters, and place all k-mers that satisfy these filters into a trie:

1. The k-mer is kept if it's error probability is below a user provided error threshold, and
2. The k-mer does not contain any IUPAC disambiguation codes.

2.2 Base Transition Probability

Due to well-known biases in the sequencing process, the four nucleotides have different probabilities of being sequenced incorrectly. Errors are known to be correlated within specific nucleotides, the position of the base in the read, and empirical quality score values. These three *covariates* are used in the GATK's base quality score recalibration (BQSR) process which measures the "true" error probability that corresponds to a base quality score [2].

[1] q-mers are k-mers which are weighted by the quality scores of the bases seen in the sequenced read.

While base quality scores are a useful prior for the GATK's main goal (specifically, variant calling), their binary error/success nature is not as useful of a prior for read error correction. Instead, since we are trying to predict whether a true base b was incorrectly sequenced as b', where $b, b' \in \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}^2$, we desire our prior to be the transition probabilities from $b' \to b$. These transition probabilities can be represented by categorical random variables with four categories. We want to estimate these distributions per error covariate, where the error covariate is defined by the sequenced base, the empirical base quality score, and the position of the base in the read.

The transition probability for error covariate $C_{b',i,b}$ can be estimated via maximum likelihood. Specifically, for each sequenced base b' at position i in the read, we can define the likelihood that this base was actually b by looking at the error probabilities of all k-mers that cover this base.

$$\mathcal{L}(b,i) = \frac{\prod_{j=1}^{k} 1 - P(\mathrm{kmer}_j^{b,i} \text{ is erroneous})}{\sum_{\hat{b}=\{A,C,G,T\}} \prod_{j=1}^{k} 1 - P(\mathrm{kmer}_j^{\hat{b},i} \text{ is erroneous})} \qquad (2)$$

In equation 2, we use $\mathrm{kmer}_j^{b,i}$ as a function that appropriately substitutes base b into the jth k-mer that covers the base at position i. Also, note in practice that bases at the start and end of the read will not be covered by k k-mers; for example, the first base of each read is covered by a single k-mer. The $P(k\text{-mer is erroneous})$ probabilities are evaluated using the trie collected by the erroneous k-mer detection process described in §2.1. We note that invalid k-mers are not present in that tree; when we search for a k-mer that is not resident in the trie, we substitute the user provided probability cutoff as an upper bound on the truth probability of that k-mer.

To achieve a most accurate estimate for the transition probabilities of error covariate $C_{b',i,b}$, we would ideally run an EM algorithm over all bases in $C_{b',i,b}$. However, this is not easily tractable as within a read, we cannot separately evaluate the transitions for bases $i-1$, i, and $i+1$. To run such an EM algorithm, we would need to evaluate all transitions for all bases in all reads, which would be computationally intractable; we discuss this in more detail in §2.3. Instead, we estimate the transition probabilities by treating the probability estimates of all bases in $C_{b',i,b}$ as the expected outcomes from multinomial trials, and apply an MLE estimator. This is equivalent to running a single expectation phase of the canonical EM algorithm for a mixture of categorical random variables.

[2] In practice, the observed base, b', may be represented by any of the IUPAC base disambiguation codes. While we handle these cases in our implementation, we do not include them here for notational simplicity. The corrected base, b, is constrained to not be a disambiguation code.

2.3 Read Refinement

3 Implementation

3.1 Distributed *k*-mer Counting

3.2 Distributed EM Algorithms

4 Results

5 Conclusion

References

1. Almhana, J., Liu, Z., Choulakian, V., McGorman, R.: A recursive algorithm for gamma mixture models. In: IEEE International Conference on Communications (ICC '06). vol. 1, pp. 197–202. IEEE (2006)
2. DePristo, M.A., Banks, E., Poplin, R., Garimella, K.V., Maguire, J.R., Hartl, C., Philippakis, A.A., del Angel, G., Rivas, M.A., Hanna, M., et al.: A framework for variation discovery and genotyping using next-generation DNA sequencing data. Nature genetics 43(5), 491–498 (2011)
3. Heo, Y., Wu, X.L., Chen, D., Ma, J., Hwu, W.M.: BLESS: Bloom filter-based error correction solution for high-throughput sequencing reads. Bioinformatics p. btu030 (2014)
4. Kelley, D.R., Schatz, M.C., Salzberg, S.L., et al.: Quake: quality-aware detection and correction of sequencing errors. Genome Biology 11(11), R116 (2010)
5. Liu, Y., Schröder, J., Schmidt, B.: Musket: a multistage k-mer spectrum-based error corrector for Illumina sequence data. Bioinformatics 29(3), 308–315 (2013)
6. Massie, M., Nothaft, F., Hartl, C., Kozanitis, C., Schumacher, A., Joseph, A.D., Patterson, D.A.: ADAM: Genomics formats and processing patterns for cloud scale computing. Tech. rep., Technical Report UCB/EECS-2013-207, EECS Department, University of California, Berkeley (2013)
7. Schwander, O., Nielsen, F.: Fast learning of gamma mixture models with k-MLE. In: Similarity-Based Pattern Recognition, pp. 235–249. Springer (2013)
8. Shi, H., Schmidt, B., Liu, W., Müller-Wittig, W.: A parallel algorithm for error correction in high-throughput short-read data on CUDA-enabled graphics hardware. Journal of Computational Biology 17(4), 603–615 (2010)
9. Song, L., Florea, L., Langmead, B.: Lighter: fast and memory-efficient error correction without counting. bioRxiv (2014)
10. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud '10). p. 10 (2010)