Improving the Accuracy of Base Calls and Error Predictions for GS 20 DNA Sequence Data

Justin S. Hogg
Department of Computational Biology
University of Pittsburgh
Pittsburgh, PA 15213
jsh32@*pitt.edu*

Abstract

New DNA sequencing technology implemented in the GS 20 sequencer reduces cost and time in exchange for lower accuracy. DNA sequencing errors negatively impact downstream applications and therefore accurate base calls and error probabilities are invaluable to researchers. This paper applies a graphical model to the base calling problem in context of the GS 20 sequencer. This model integrates signal information from the local sequence neighborhood to generate calls within a probabilistic framework. Results indicate improved accuracy for base calls early in the sequence process, but the overall perforance decreases.

1 Introduction

The GS 20 is a DNA sequencer based on new technology developed by 454 Life Sciences. The platform is massively parallel, fast and cost effective in comparison to Sanger technology. The trade off is a relatively high nucleotide base calling error rate of ~1 error per 100 bases (after discarding low quality sequence). Sequence accuracy can be improved by sequencing the same DNA template multiple times, but this is not always economically feasible for large genome sequencing projects. Sequencing errors have severe negative impact on downstream applications, such as gene identification, genotyping and mutation detection. The ideal error rate—less than one error in ten thousand bases—is unlikely to be achieved. In lieu of higher accuracy, error probabilities are invaluable to researchers.

The GS 20 sequences DNA by a query and response process. In each step the sequencer asks: "How many consecutive T nucleotides (A,C,G respectively) appear next in the sequence?" The DNA responds by producing a light signal with intensity proportional to the number of T's. The same query is made iteratively for each of the four DNA bases, and the cycle is repeated. Each base query is called a "flow". The cycle of queries T,A,C,G is called a "flow cycle". Each sequencing run consists of 42 flow cycles (168 flows), and is known as a "flowgram". See the publication in *Nature* for technical details of the physical sequencing process [1].

This process is susceptible to several types of sequencing errors. Here I discuss the error types that are relevant to this paper. The first type of error results from an increase in the noise variance as the signal intensity increases. Consequently base calls tend to be accurate when a single base is measured, while accuracy decreases rapidly when a single base type occurs consecutively many times in a sequence. The second type of error results from a loss of synchronization. This is due to the fact that the sequencing reaction is performed simultaneously on many thousands of identical copies of the DNA template. The combined signal from the cloned templates produces sufficient light signal for detection by CCD cameras. If a fraction of the templates are de-synchronized, then the resulting signal is a sum of the current query and the previous and next queries (of the same base type). This problem increases toward the end of the sequence and is most severe when the previous or next flow of the same base type measures a large signal. The final type of error is due to residual signal from other flows. This problem is also most severe when the previous or next query (of any base type) measures a large signal.

454 Life Sciences has developed propriety software which processes image data, normalizes signals, predicts DNA base calls, and reports a quality score for each base. The quality score has the form $Q = -10*log_{10} P(Error)$ where $P(Error)$ is the probability that the base call is incorrect. $P(Error)$ is calculated using Bayes' rule from experimentally determined signal distributions. This method does not incorporate information from nearby flow signals which may provide evidence that increases the probability of a base call error. The following work attempts to maximize the accuracy of base calls and error predictions by incorporating signal information from the local neighborhood.

2 A Graphical Model representation of the GS 20 flowgram

The base calling problem is represented by a graphical model where the observed variables are 168 consecutive flow signals, **S**, measured by the GS 20 in each sequencing reaction. The latent variables are 168 base calls **B**, corresponding to the 168 flow signals, where the base call is the number of consecutive type A nucleotides (G, C, or T respectively) at the current position in the sequence. The nth flow signal measurement, S_n, is dependent primarily on base call B_n and also on neighboring base calls. In particular, the flow signal is dependent on B_{n-4} and the B_{n+4} to model the effect of de-synchronization, and B_{n-1} to model the effects of residual signal from the previous flow. This formulation is computationally difficult since the observation of flow signals allows information passing between the base variables through the head-to-head path at the observed signal variable. Consequently all 168 signal variables are conditionally dependent. This problem can be resolved by "flipping" the arrows in the model so the base variables are dependent on the signal variables. In this formulation the observed variables block the flow of information at a tail-to-tail path. Consequently the probability distribution for B_n is only dependent 4 observed flow signals. Physically this formulation is backwards, but in probabilistic framework there is obstacle to modeling the process in this manner. This reduces the graphical model to a set of 168 independent discrete probability distributions $P(B_n|S_{n-4}, S_{n-1}, S_n, S_{n+4})$ which can be learned by multi-class logistic regression. The distribution of B_n is similar for neighboring flows. Consequently, neighboring flows can be grouped together and using to train a single model for the whole group. However, since the signal noise increases over the duration of the sequence procedure, it is important to divide the flowgram into a number of groups which are trained independently to capture position dependent effects.

The basecall B_n may, in practice, take any integer value ≥ 0. This poses a challenge during parameter estimation since consecutive repeats of a single base longer than 4 bases is uncommon in a DNA sequence. This problem is circumvented using two methods. First, all base calls greater than *k-1* are grouped into a single category. Under this scheme B_n is described by a *1-of-k* encoding (corresponding to base calls of *0* to *k-1*). This simplification prevents the model from distinguishing between base calls of *k-1* or larger, but avoids the pitfalls of over fitting sparse data. Secondly, during the training phase examples are weighted in a class dependent fashion. Common classes are assigned a unit weight, while uncommon classes are weighted more heavily. The weights effectively add "extra" training examples from uncommon classes. This counteracts the tendency to lump the uncommon examples in a more common class.

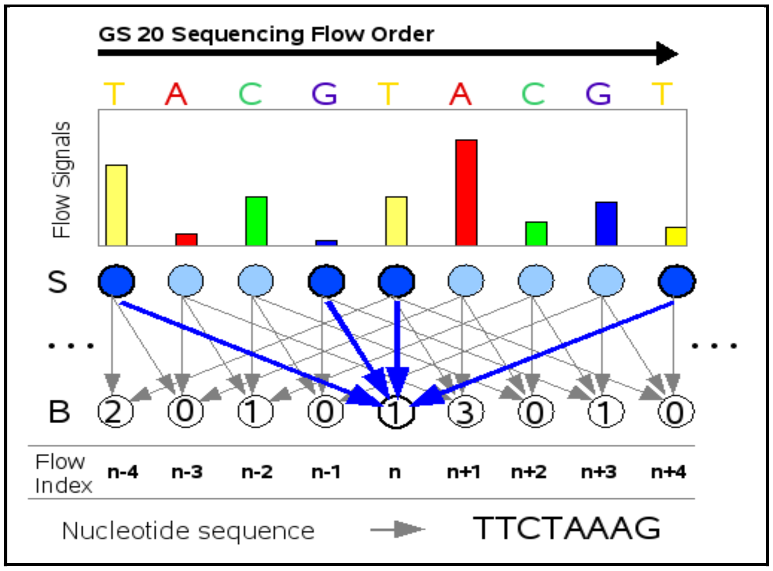

Figure 1: A graphical model representation of GS 20 sequence data.

3 Methods

The model was trained using *~36,000* normalized flowgrams generated by the *GS 20* using *P.aeruginosa* strain PA01 genomic DNA as template. The base calls for each flowgram were initially generated by 454 Life Sciences proprietary software and then aligned to the high quality published PA01 genome by the MUMmer software package. The sequence alignments were used to label the training data. The labeling process was automated using a Perl script. *36,000* additional PA01 flowgrams were set aside for test validation.

Flows were divided into 19 groups containing 8 consecutive flows. A single classifier was trained for each group. The flows were divided in this manner to account for the gradual increase in the signal noise towards the end of sequencing process. The first 8 control flows were omitted from the analysis, as well as the first and last four flows of each flowgram (to avoid flows without all relevant neighbors). The form of $P(B_n | S_{n-4}, S_{n-1}, S_n, S_{n+4})$ is modeled by a weighted Logistic Regression where B_n is a *1-of-5* encoding of the base call, and each S_n is a continuous variable. Training examples were weighted according to their base class to account for sparsity of training examples with class with $k>2$. Class weights {*1, 1 , 3, 12, 42*} were selected for the respective 5 classes by training classifiers with a variety of weights and choosing those which produced a reasonable balance of accuracy across all classes without significantly compromising overall accuracy. Regression parameters were trained using a gradient descent algorithm with regularization implemented in MATLAB as:

$$\boldsymbol{W}^{(n)} \leftarrow \boldsymbol{W}^{(n)} + \eta \boldsymbol{E}^{(n)} \boldsymbol{X}^{(n)} - \eta \lambda \boldsymbol{W}^{(n)}$$

Where n is index of the flow group, w_{ij} is the weight parameter for class j of signal input i, $\eta=1.0e\text{-}6$ is the step size, $\lambda=0.01$ is the regularization parameter, e_{jl} is the weighted error of class j for training example l, and x_{li} is signal input i for training example l. The weighted error **E** is defined by:

$$\boldsymbol{E}=\boldsymbol{C}\cdot[\boldsymbol{Y}-P(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{W})]^T$$

Where **C** is a diagonal matrix of class weights, **Y** is the matrix whose rows are the *1-of-5* encoding of the true base calls, and P(**Y**|**X**,**W**) is the matrix of class probabilities. Optimization of the regularization parameter by validation over the test set is planned for the final project submission. Dependence of the results on regularization was negligible. This was expected due to the large number of training examples in comparison to the number of parameters. Base calls were generated by choosing the class with the highest probability.

4 Results

The model trained with uniform class weights was very accurate for classes $k=0$ and $k=1$. However, accuracy dropped rapidly for classes $k\geq 2$ due to the lower frequency of these classes. Table 1 shows the effects of various class weights on the accuracy of each class. In all cases the accuracy decreases in the larger classes due to increasing signal variance. The weights {*1,1,3,12,42*} were chosen for the final analysis as a balance between overall accuracy and improved discrimination for the larger classes. The size of the weights is similar to, but slightly less than, the relative frequency of the base classes.

Table 1: Class error rates for group 1 using various class weights. Unit weights result in 100% error for base calls 3 or greater. Increasing weights for underrepresented classes improves the performance in these categories, but at the expense of some accuracy for base calls 2 and lower.

CLASS WEIGHTS	ERROR K=0	ERROR K=1	ERROR K=2	ERROR K=3	ERROR K=4+
1,1,1,1,1	.0004	.0003	.0478	1.000	1.000
1,1,4,16,64	.0006	.0020	.0025	.1271	1.000
1,1,3,12,42	.0006	.0011	.0036	.3872	.4870
1,1,3,12,48	.0006	.0011	.0036	.5052	.1005
1,1,3,9,27	.0006	.0011	.0029	.5098	.9536
1,1,2,8,16	.0005	.0005	.0122	.2607	1.000

The average error rate of the final model on the test data set was 2.49%. However, the accuracy depends on several factors, and in many cases the error is much lower or higher than average. The first factor is the position of the flow within the flowgram. The average error for a base call in the first half of the flowgram is 0.87%, while the error in the second half increases to 4.0%. This trend emphasizes the importance of training separate models along the length of the flowgram. The error is also dependent on the base class. The average error rates for each class are: 0.39%, 0.66%, 1.1%, 14.3%, and 13.9% (respectively) for the first half of the flowgram. The model is ineffective at determining the exact number of consecutive nucleotides when there are 3 or more of the same type. Figure 2 summarizes these results.

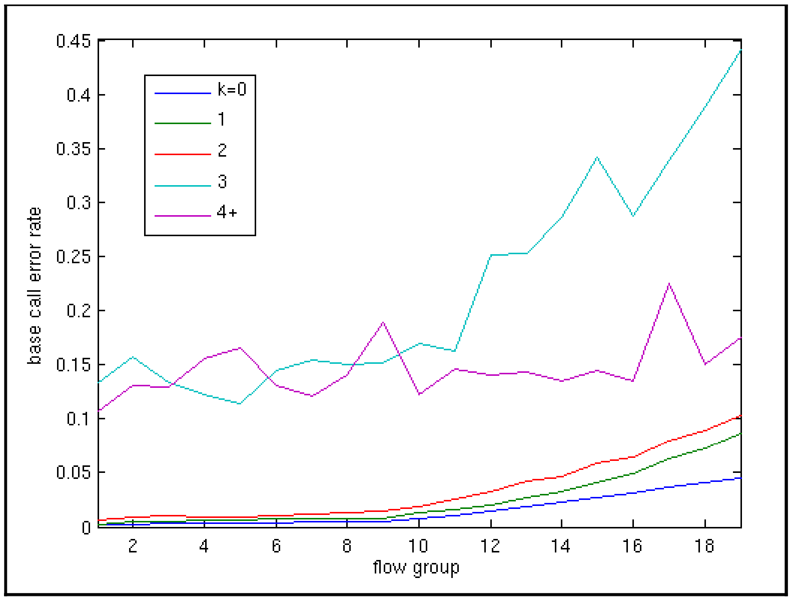

Figure 2: Base call error rate for each group and class. The error rate in the first half of the flowgram is near 1% for basecalls of 0, 1 and 2; but much higher for 3 and 4+. Error rates increase significantly toward the end of the flowgram, as expected.

In order to determine the effectiveness of this approach, a simple model using only 1 signal input per flow was trained on the same data set. A comparison of the performance between the full model and the single input model is shown in figure 3. The overall error rate of the single input model is lower than the full model (2.39% error). However, the full model performs better in the first half of the flowgram.

5 Discussion

Integrating signal information from the neighborhood surrounding a flow yields a modest increase in the accuracy of base calls in the early portion of a flowgram. This result agrees with the observation that signals generated by the GS 20 are dependent on the nature of the local sequence. However, the overall accuracy of this approach is lower than a simple model that considers only the primary signal. This calls into question the utility of the method. It appears that the information available in the neighboring signals is small and easily overwhelmed by the increase in signal variance towards the end of the flowgram. A combination of the two models, where the second half of the flowgram is evaluated by the single input model, may yield the best overall results.

The GS 20, combined with methods here, is fairly accurate in discriminating between base calls 0 and 1 in the first half of a flowgram (average error 0.5%). This suggests that GS 20 data is most reliable for single nucleotide polymorphisms (SNPs) application when the neighboring bases are distinct from the SNP in question. Other applications, such as ORF identification, are likely to be problematic.

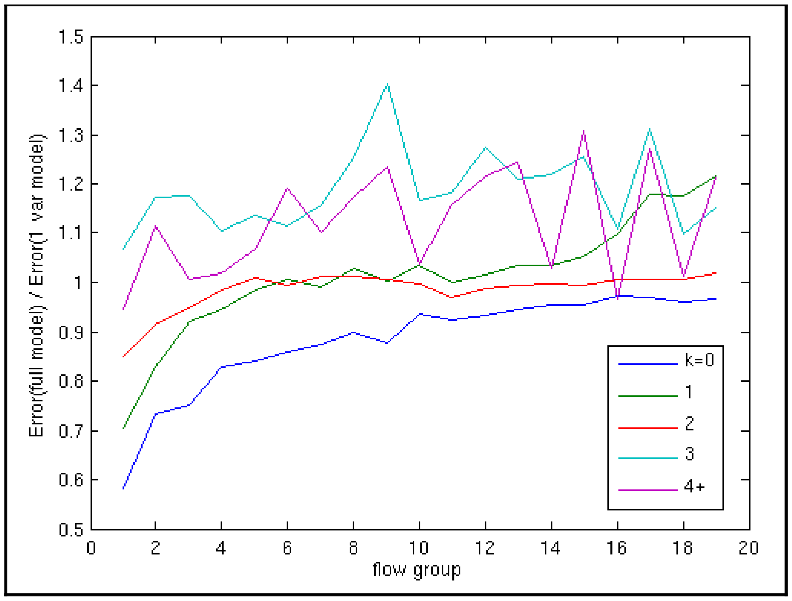

Figure 3: Ratio of base call error rate for the full model over a model trained using only one signal input. The full model performs better for classes *k=0,1,2* in the first half of the flowgram. The single input model tends to perform better for classes *k=3,4+* and in the second half of the flowgram.

Acknowledgments

Access to PA01 sequence data generated on the *GS 20* was provided by the *Center for Genomic Sciences*, Pittsburgh PA.

References

[1] Margulies, et.al. (2005) Genome Sequencing in microfabricated high-density picolitre reactors. *Nature* 437():376-380.

[2] GS 20 Data Processing Software Manual. Section 4: Signal Processing. Roche Applied Sciences.