naiveBayesCall: An efficient model-based base-calling algorithm for high-throughput sequencing

Wei-Chun Kao[1] and Yun S. Song[1,2]

[1] Computer Science Division, University of California, Berkeley, CA 94720, USA
[2] Department of Statistics, University of California, Berkeley, CA 94720, USA
`wckao@eecs.berkeley.edu, yss@eecs.berkeley.edu`

Abstract. Immense amounts of raw instrument data (i.e., images of fluorescence) are currently being generated using ultra high-throughput sequencing platforms. An important computational challenge associated with this rapid advancement is to develop efficient algorithms that can extract accurate sequence information from raw data. To address this challenge, we recently introduced a novel model-based base-calling algorithm that is fully parametric and has several advantages over previously proposed methods. Our original algorithm, called BayesCall, significantly reduced the error rate, particularly in the later cycles of a sequencing run, and also produced useful base-specific quality scores with a high discrimination ability. Unfortunately, however, BayesCall is too computationally expensive to be of broad practical use. In this paper, we build on our previous model-based approach to devise an efficient base-calling algorithm that is orders of magnitude faster than BayesCall, while still maintaining a comparably high level of accuracy. Our new algorithm is called naive-BayesCall, and it utilizes approximation and optimization methods to achieve scalability. We describe the performance of naiveBayesCall and demonstrate how improved base-calling accuracy may facilitate de novo assembly when the coverage is low to moderate.

1 Introduction

Recent advances in sequencing technology is enabling fast and cost-effective generation of sequence data, and complete whole-genome sequencing will soon become a routine part of biomedical research. The key feature of the next-generation sequencing technology is parallelization and the main mechanism underlying several platforms is sequencing-by-synthesis (SBS); we refer the reader to [1, 15] for a more comprehensive introduction to SBS and whole-genome re-sequencing. Briefly, tens to hundreds of millions of random DNA fragments get sequenced simultaneously by sequentially building up complementary bases of single-stranded DNA templates and by capturing the synthesis information in a series of raw images of fluorescence. Extracting the actual sequence information (i.e., strings in {A, C, G, T}) from image data involves two computational problems, namely image analysis and base-calling. The primary function of image analysis is to translate image data into fluorescence intensity data for each DNA fragment, while the goal of base-calling is to infer sequence information from the obtained intensity data. Although algorithms developed by the manufacturers of the next-generation sequencing platforms work reasonably well, it is widely recognized that independent researchers must develop improved algorithms for optimizing data acquisition, to reduce the error rate and to reduce the cost of sequencing by increasing the throughput per run.

At present, Illumina's Genome Analyzer (GA) is the most widely-used system among the competing next-generation sequencing platforms. In GA, SBS is carried out on a glass surface called the flow cell, which consists of 8 lanes, each with 100 tiles. In a typical sequencing run, each tile holds about a hundred thousand clusters, with each cluster containing about 1000 identical DNA templates. The overall objective is to infer the sequence information for each cluster. The base-calling software supplied with GA is called Bustard, which adopts a very efficient algorithm based on matrix inversion. Although the algorithm works very well for the early cycles of a sequencing run, it is well-known that the error rate of Bustard becomes substantial in later cycles. Reducing the error rate of base-calls and improving the accuracy of base-specific quality score will have important practical implications for assembly [3,4,11,12,14,17,21], polymorphism detection (especially rare ones) [2,12], and downstream population genomics analysis of next-generation sequencing data [7,8].

Recently, several improved base-calling algorithms [5,9,16,19] have been developed for the Illumina platform. In particular, a large improvement in accuracy was achieved by our own method called BayesCall [9]. The key feature that distinguishes BayesCall from the other methods is the explicit modeling of the sequencing process. In particular, BayesCall explicitly takes residual effects into account and is the only existing base-calling algorithm that can incorporate time-dependent parameters. Importantly, parameter estimation is done *unsupervised* and BayesCall produces very good results even when using a very small training set consisting of only a few hundred randomly chosen clusters. This feature enables the estimation of local parameters to account for the potential differences between different tiles and lanes. Furthermore, being a fully parametric model, our approach provides information on the relative importance of various factors that contribute to the observed intensities, and such information may become useful for designing an improved sequencing technology.

Supervised machine learning is an alternative approach that other researchers have considered in the past for base-calling. For example, Alta-Cyclic [5] is a method based on the support vector machine that requires a large amount of labeled training data. To create a rich training library, in every sequencing run it requires using a control lane containing a sample with a known reference genome. Note that using such a control incurs cost and takes up space on the flow cell that could otherwise be used to sequence a sample of interest to the biologist. Furthermore, this approach cannot handle variability across lanes.

In [9], we showed that our method significantly improves the accuracy of base-calls, particularly in the later cycles of a sequencing run. In addition, we showed that BayesCall produces quality scores with a high discrimination ability [6] that consistently outperforms both Bustard's and Alta-Cyclic's. Unfortunately, however, this improvement in accuracy came at the price of substantial increase in running time. BayesCall is based on a generative model and performs base-calls by maximizing the posterior distribution of sequences given observed data (i.e., fluorescence intensities). This step involves using the Metropolis-Hastings algorithm with simulated annealing, which is computationally expensive; it would take several days to base-call a single lane using a desktop computer. This slow running time seriously restricts the practicality of BayesCall.

The goal of this paper is to build on the ideas behind BayesCall to devise an efficient base-calling algorithm that is orders of magnitude faster than BayesCall, while still maintaining a comparably high level of accuracy. There are two computational parts to BayesCall: parameter estimation and base-calling. Since estimation of the time-dependent parameters in BayesCall can be performed progressively as the sequencing machine runs, we believe that the bottleneck is in the base-calling part. Our new algorithm is called naiveBayesCall. It is based on the same generative model as in BayesCall and employs the same parameter estimation method as before (see [9] for details). However, in contrast to BayesCall, our new algorithm avoids doing Markov chain Monte Carlo sampling in the base-calling part of the algorithm. Instead, naiveBayesCall utilizes approximation and optimization methods to achieve scalability. To test the performance of our method, we use a standard resequencing data of PhiX174 virus, obtained from a 76-cycle run on Illumina's GA II platform. Then, we demonstrate how improved base-calling accuracy may facilitate de novo assembly.

Our software implementation can be run either on an ordinary PC or on a computing cluster, and is fully compatible with the file formats used by Illumina's GA pipeline. Our software is available at http://bayescall.sourceforge.net/.

2 A review of the model underlying the original BayesCall algorithm

Our main goal in developing BayesCall was to model the sequencing process in GA to the best of our knowledge, by taking stochasticity into account and by explicitly modeling how errors may arise. In each cycle, ideally the synthesis process is supposed to add exactly one complementary base to each template, but, unfortunately, this process is not perfect and some templates may jump ahead (called prephasing) or lag behind (called phasing) in building up complementary strands. This is a major source of complication for base-calling. Environment factors such as temperature fluctuation also contribute to stochasticity.

Below, we briefly review the main ideas underlying BayesCall [9]. Throughout, we adopt the same notational convention as in [9]: Multi-dimensional variables are written in boldface, while scalar variables are written in normal face. The transpose of a matrix $\boldsymbol{M}$ is denoted by $\boldsymbol{M}'$. The index t is used to refer to a particular cycle, while the index k is used to refer to a particular cluster of identical DNA templates. The total number of cycles in a run is denoted by L.

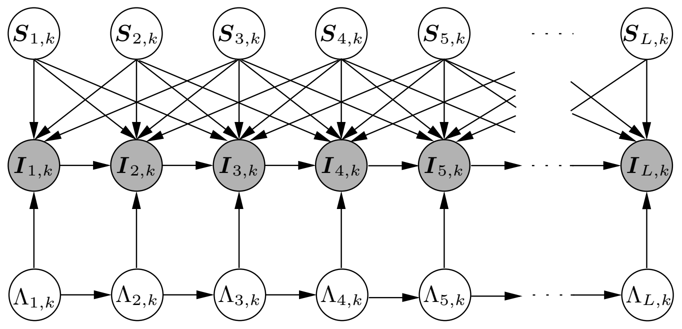

Fig. 1. The graphical model for BayesCall. The observed random variables are the intensities $\boldsymbol{I}_{t,k}$. Base-calling is done by finding the maximum a posteriori estimates of $\boldsymbol{S}_{t,k}$. In this illustration, the window within which we consider phasing and prephasing effects has size 5. In our implementation, we use a window of size 11.

Let $\boldsymbol{e}_i$ denote a 4-component column unit vector with a 1 in the ith entry and 0s elsewhere. We use the basis with A, C, G, T corresponding to indices $1, 2, 3, 4$, respectively.

BayesCall is founded on a graphical model, illustrated in Figure 1. It involves the following random variables:

Sequence ($\boldsymbol{S}_k$): We use $\boldsymbol{S}_k = (\boldsymbol{S}_{1,k}, \ldots, \boldsymbol{S}_{L,k})$, with $\boldsymbol{S}_{t,k} \in \{\boldsymbol{e}_A, \boldsymbol{e}_C, \boldsymbol{e}_G, \boldsymbol{e}_T\}$, to denote the 4-by-$L$ binary *sequence matrix* corresponding to the complementary sequence of the DNA templates in cluster k. The main goal of base-calling is to infer $\boldsymbol{S}_k$ for each cluster k. We assume a uniform prior on sequences:

$$\boldsymbol{S}_{t,k} \sim \mathrm{Unif}(\boldsymbol{e}_A, \boldsymbol{e}_C, \boldsymbol{e}_G, \boldsymbol{e}_T).$$

If the genome-wide nucleotide distribution of the sample is known, then that distribution may be used here instead, which should improve the accuracy of base-calls.

Active template density ($\mathit{\Lambda}_{t,k}$): In BayesCall, fluctuation in fluorescence intensity over time is explicitly modeled using a random variable $\mathit{\Lambda}_{t,k}$ that corresponds to the per-cluster density of templates that are "active" (i.e., able to synthesize further) at cycle t in cluster k. Given $\mathit{\Lambda}_{t-1,k}$ from the previous cycle, $\mathit{\Lambda}_{t,k}$ is distributed as a 1-dimensional normal distribution with mean $(1-d_t)\mathit{\Lambda}_{t-1,k}$ and variance $(1-d_t)^2\mathit{\Lambda}^2_{t-1,k}\sigma_t^2$:

$$\mathit{\Lambda}_{t,k} | \mathit{\Lambda}_{t-1,k} \sim \mathcal{N}((1-d_t)\mathit{\Lambda}_{t-1,k}, (1-d_t)^2\mathit{\Lambda}^2_{t-1,k}\sigma_t^2). \tag{1}$$

Intensities ($\boldsymbol{I}_{t,k}$): We use $\boldsymbol{I}_{t,k} = (I^A_{t,k}, I^C_{t,k}, I^G_{t,k}, I^T_{t,k})' \in \mathbb{R}^{4\times 1}$ to denote the fluorescence intensities of A, C, G, T channels at cycle t in cluster k, after subtracting out the background signals. These are the observed random variables in our graphical model.

As mentioned above, the DNA synthesis process is not perfect and may go out of "phase." In Bustard and BayesCall, the synthesis process in a cycle is modeled by a Markov model in which the position of the terminating complementary nucleotide of a given template changes from i to j according to the transition matrix $\boldsymbol{P} = (P_{ij})$ given by

$$P_{ij} = \begin{cases} p, & \text{if } j = i, \\ 1-p-q, & \text{if } j = i+1, \\ q, & \text{if } j = i+2, \\ 0, & \text{otherwise,} \end{cases}$$

where $0 \leq i, j \leq L$. Here, p denotes the probability of phasing (i.e., no new base is synthesized during the cycle), while q denotes the probability of prephasing (i.e., two bases are synthesized). Normal synthesis of a single complementary nucleotide occurs with probability $1 - p - q$. At cycle 0, we assume that all templates start at position 0; i.e., no nucleotide has been synthesized. Note that the (i, j) entry of the matrix $\boldsymbol{P}^t$ corresponds to the probability that a terminator at position i moves to position j after t cycles.

Define Q_{jt} as the probability that a template terminates at position j after t cycles. It is easy to see that $Q_{jt} = [\boldsymbol{P}^t]_{0,j}$, the $(0, j)$ entry of the matrix $\boldsymbol{P}^t$. We use $\boldsymbol{Q}_t$ to denote column t of the L-by-L matrix $\boldsymbol{Q} = (Q_{jt})$. In practice, $\boldsymbol{Q}_t$ will have only a few dominant components, with the rest being very small. More precisely, the dominant components will be concentrated about the tth entry. Therefore, at cycle t, we simplify the computation by considering phasing and prephasing effects only within a small window w about position t. Let $\boldsymbol{Q}_t^w$ denote the L-dimensional column vector obtained from $\boldsymbol{Q}_t$ by setting the entries outside the window to zero. Hence, the concentration of active templates in cluster k with A, C, G, T terminating complementary nucleotide can be approximated by the following 4-dimensional vector:

$$\boldsymbol{Z}_{t,k}^w = \Lambda_{t,k} \boldsymbol{S}_k \boldsymbol{Q}_t^w. \tag{2}$$

The four fluorophores used to distinguish different terminating nucleotides have overlapping spectra [20], and this effect can be modeled as $\boldsymbol{X}_t \boldsymbol{Z}_{t,k}^w$, where $\boldsymbol{X}_t \in \mathbb{R}^{4\times4}$ is a matrix called the *crosstalk* matrix, with $(X_t)_{ij}$ denoting the response in channel i due to fluorescence of a unit concentration of base j. (We refer the reader to [13] for discussion on estimating $\boldsymbol{X}_t$.)

In addition to phasing and prephasing effects, we observed other residual effects that propagate from one cycle to the next. We found that modeling such extra residual effects improves the base-call accuracy. In BayesCall, we introduced parameters α_t and assumed that the observed intensity $\boldsymbol{I}_{t,k}$ at cycle t contains the residual contribution $\alpha_t(1 - d_t)\boldsymbol{I}_{t-1,k}$ from the previous cycle. In summary, the mean fluorescence intensity for cluster k at cycle t is given by

$$\boldsymbol{\mu}_{t,k} = \boldsymbol{X}_t \boldsymbol{Z}_{t,k}^w + \alpha_t(1 - d_t)\boldsymbol{I}_{t-1,k}, \tag{3}$$

with $\boldsymbol{I}_{0,k}$ defined as the zero vector $\mathbf{0}$. Finally, with the assumption that the background noise at cycle t is distributed as Gaussian white noise with zero mean and covariance matrix $\|\boldsymbol{Z}^w_{t,k}\|^2 \boldsymbol{\Sigma}_t$, where $\|\cdot\|$ denotes the 2-norm, the observed fluorescence intensity in BayesCall is distributed as the following 4-dimensional normal distribution:

$$\boldsymbol{I}_{t,k} | \boldsymbol{I}_{t-1,k}, \boldsymbol{S}_k, \Lambda_{t,k} \sim \mathcal{N}(\boldsymbol{\mu}_{t,k}, \|\boldsymbol{Z}^w_{t,k}\|^2 \boldsymbol{\Sigma}_t), \tag{4}$$

where the mean is shown in (3).

In BayesCall, global parameters p, q, and cycle-dependent parameters $d_t, \alpha_t, \sigma_t, \boldsymbol{X}_t, \boldsymbol{\Sigma}_t$ are estimated using the expectation-maximization (EM) algorithm, combined with Monte-Carlo integration via the Metropolis-Hastings algorithm.

3 naiveBayesCall: a new algorithm

We now describe our new algorithm naiveBayesCall. As mentioned in Introduction, it is based on the same graphical model as in BayesCall, and we employ the method detailed in [9] to estimate the parameters in the model. The main novelty of naiveBayesCall is in the base-calling part of the method. We divide the presentation of our new base-calling algorithm into two parts. First, we propose a hybrid algorithm that combines the model described in Section 2 with the matrix inversion approach employed in Bustard. Then, we use the hybrid algorithm to initialize an optimization procedure that both improves the base-call accuracy and produces useful per-base quality scores.

3.1 A hybrid base-calling algorithm

We present a new inference algorithm for the model described in Section 2. The main strategy is to avoid direct inference of the continuous random variables $\Lambda_{t,k}$. First, for each cycle t, we estimate the average concentration c_t of templates within each tile. In [9], we showed that the magnitude of the fluctuation rate d_t (c.f., (1)) is typically very small (less than 0.03) for all $1 \le t \le L$. Hence, assuming that d_t is close to zero for all t, we estimate the tile-wide average concentration c_t using

$$c_t = \frac{1}{K} \sum_{k=1}^{K} \sum_{b=1}^{4} \max(0, [\boldsymbol{X}_t^{-1}(\boldsymbol{I}_{t,k} - \alpha_t \boldsymbol{I}_{t-1,k})]_b), \tag{5}$$

where K denotes the total number of clusters in the tile and $[\boldsymbol{y}]_b$ denotes the bth component of vector $\boldsymbol{y}$. The above c_t serves as our estimate of $\Lambda_{t,k}$ for all clusters k within the same tile. Using this estimate, we define

$$\tilde{\boldsymbol{I}}_{t,k} = \left(\boldsymbol{I}_{t,k} - \alpha_t \frac{c_t}{c_{t-1}} \boldsymbol{I}_{t-1,k} \right)_+, \tag{6}$$

where $(\boldsymbol{y})_+$ denotes the vector obtained from $\boldsymbol{y}$ by replacing all negative components with zeros. Note that subtracting $\alpha_t \frac{c_t}{c_{t-1}} \boldsymbol{I}_{t-1,k}$ from $\boldsymbol{I}_{t,k}$ accounts for the

Algorithm 1 Hybrid Algorithm

```
for all tiles do
  for all cycles $1 \le t \le L$ do
    Estimate concentration $c_t$ for each cycle $t$ according to (5).
  end for
  for all clusters $1 \le k \le K$ do
    Compute residual-corrected intensities $\tilde{\boldsymbol{I}}_k$ using (6).
    Compute concentration matrix $\boldsymbol{\mathcal{Z}}_k$ according to (7).
    Correct for phasing and prephasing effect using (8).
    Infer $\boldsymbol{S}^H_{1,k}, \ldots, \boldsymbol{S}^H_{L,k}$ using (9) and output the associated sequence.
  end for
end for
```

residual effect modeled in (3). The ratio $\frac{c_t}{c_{t-1}}$ rescales $\boldsymbol{I}_{t-1,k}$ so that its norm is similar to that of $\boldsymbol{I}_{t,k}$.

After determining $\tilde{\boldsymbol{I}}_{t,k}$, the rest of the hybrid base-calling algorithm resembles Bustard. (For a detailed description of Bustard, see [9].) First, for each cycle t, we estimate the cluster-specific normalized concentration of four different bases using

$$\boldsymbol{\mathcal{Z}}_{t,k} = (\mathcal{Z}^A_{t,k}, \mathcal{Z}^C_{t,k}, \mathcal{Z}^G_{t,k}, \mathcal{Z}^T_{t,k})' = \frac{1}{c_t}(\boldsymbol{X}_t^{-1}\tilde{\boldsymbol{I}}_{t,k})_+, \tag{7}$$

where $\boldsymbol{X}_t$ is the 4-by-4 crosstalk matrix at cycle t (see previous section). Normalizing by the tile-wide average c_t is to make the total concentration stay roughly the same across all cycles. Note that $\boldsymbol{\mathcal{Z}}_{t,k}$ is an estimate of the concentration vector shown in (2). Now, we let $\boldsymbol{\mathcal{Z}}_k = (\boldsymbol{\mathcal{Z}}_{1,k}, \ldots, \boldsymbol{\mathcal{Z}}_{L,k})$ and use the following formula to correct for phasing and prephasing effects:

$$\boldsymbol{\mathcal{Z}}_k(\boldsymbol{Q}^w)^{-1}, \tag{8}$$

where $\boldsymbol{Q}^w = (\boldsymbol{Q}^w_1, \ldots, \boldsymbol{Q}^w_L)$ is the L-by-L phasing-prephasing matrix defined in Section 2. Finally, for $t = 1, \ldots, L$, the row index of the largest value in column t of (8) is called as the tth base of the DNA templates in cluster k:

$$\boldsymbol{S}^H_{t,k} = \underset{b\in\{A,C,G,T\}}{\operatorname{argmax}} [\boldsymbol{\mathcal{Z}}_k(\boldsymbol{Q}^w)^{-1}]_{b,t}. \tag{9}$$

Algorithm 1 summarizes the hybrid base-calling algorithm just described.

The performance of the hybrid algorithm will be discussed in Section 4. We will see that, with the parameters estimated in BayesCall, our simple hybrid algorithm already outperforms Bustard.

3.2 Estimating $\boldsymbol{\Lambda}_k$ via optimization and computing quality scores

In this section, we devise a method to improve the hybrid algorithm described above and to compute base-specific quality score. The Viterbi algorithm [18] has been widely adopted as a dynamic programming algorithm to find the most probable path of states in a hidden Markov Model. There are two source of difficulty in applying the Viterbi algorithm to our problem:

Algorithm 2 naiveBayesCall Algorithm

```
for all clusters $k$ do
    Initialize $\boldsymbol{S}_k^{(0)} = (\boldsymbol{S}_{1,k}^H, \ldots, \boldsymbol{S}_{L,k}^H)$ using Algorithm 1.
    for $1 \leq t \leq L$ do
        for $b \in \{A, C, G, T\}$ do
            Find $\lambda_{t,k}^b = \mathrm{argmax}_\lambda L_{t,k}^b(\lambda)$, where $L_{t,k}^b(\lambda)$ is defined as in (10).
            Compute base-specific quality score $Q(b)$ using (14) and (15).
        end for
        Set $s_{t,k} = \mathrm{argmax}_{b \in \{A,C,G,T\}} L_{t,k}^b(\lambda_{t,k}^b)$.
        Update $\boldsymbol{S}_k^{(t)} = \mathcal{R}_{t,s_{t,k}}(\boldsymbol{S}_k^{(t-1)})$.
    end for
    Call $s_{1,k}, \ldots, s_{L,k}$ as the inferred sequence and output base-specific quality scores.
end for
```

1. Our model is a high order Markov model, so path tracing can be computationally expensive. This complexity arises from modeling phasing and prephasing effects. Recall that the observation probability at a given cycle t depends on all hidden random variables $\boldsymbol{S}_{i,k}$ with i within a window w about t. In [9], we used 11 for the window size.
2. In addition to the discrete random variables $\boldsymbol{S}_k = (\boldsymbol{S}_{1,k}, \ldots, \boldsymbol{S}_{L,k})$ for the DNA sequence, our model contains continuous hidden random variables $\boldsymbol{\Lambda}_k = (\Lambda_{1,k}, \ldots, \Lambda_{L,k})$, but the Viterbi algorithm cannot handle continuous variables. One might try to address this problem by marginalizing out $\boldsymbol{\Lambda}_k$, but it turns out that the maximum a posteriori (MAP) estimate of $\boldsymbol{\Lambda}_k$ is useful for computing quality scores.

To address the first problem, we obtain a good initial guess of hidden variables $\boldsymbol{S}_k$ and use it to break the high order dependency. To cope with the second problem, we adopt a sequential approach. Algorithm 2 summarizes our naiveBayesCall algorithm and a detailed description is provided below.

Our algorithm iteratively estimates $\Lambda_{t,k}$ and updates $\boldsymbol{S}_{t,k}$, starting with $t = 1$ and ending at $t = L$. Let $\boldsymbol{S}_k^{(i)}$ denote the sequence matrix after the ith iteration. We initialize $\boldsymbol{S}_k^{(0)} = \boldsymbol{S}_k^H$, where $\boldsymbol{S}_k^H = (\boldsymbol{S}_{1,k}^H, \ldots, \boldsymbol{S}_{L,k}^H)$ is obtained using the hybrid algorithm described in Section 3.1. Let $s_{t,k}$ denote the base (i.e., A, C, G, or T) called by naiveBayesCall for position t of the DNA sequence in cluster k. At iteration t, the first $t-1$ bases $s_{1,k}, \ldots, s_{t-1,k}$ have been called and the vectors $\boldsymbol{S}_{1,k}, \ldots, \boldsymbol{S}_{t-1,k}$ have been updated accordingly. The following procedures are performed at iteration t:

Optimization: Our inference of $\Lambda_{t,k}$ depends on the base at position t, which has not been called yet. We use $\lambda_{t,k}^b$ to denote the inferred value of $\Lambda_{t,k}$, given that the base at position t is b. For a given base $b \in \{A, C, G, T\}$, we define the log-likelihood function

$$L_{t,k}^b(\lambda) = \begin{cases} \log \mathbb{P}[\boldsymbol{I}_{t,k} | \boldsymbol{I}_{t-1,k}, \mathcal{R}_{t,b}(\boldsymbol{S}_k^{(t-1)}), \lambda], & \text{if } t = 1, \\ \log \mathbb{P}[\lambda | \lambda_{t-1,k}^{s_{t-1,k}}] + \log \mathbb{P}[\boldsymbol{I}_{t,k} | \boldsymbol{I}_{t-1,k}, \mathcal{R}_{t,b}(\boldsymbol{S}_k^{(t-1)}), \lambda], & \text{if } t > 1, \end{cases} \tag{10}$$

where $\mathcal{R}_{t,b}(\boldsymbol{S}_k^{(t-1)})$ denotes the sequence matrix obtained by replacing column t of $\boldsymbol{S}_k^{(t-1)}$ with the unit column vector $\boldsymbol{e}_b$, the probability $\mathbb{P}[\lambda|\lambda_{t-1,k}^{s_{t-1,k}}]$ is defined in (1), and observation likelihood $\mathbb{P}[\boldsymbol{I}_{t,k}|\boldsymbol{I}_{t-1,k}, \mathcal{R}_{t,b}(\boldsymbol{S}_k^{(t-1)}), \lambda]$ is defined by (2)–(4), More exactly,

$$\mathbb{P}[\boldsymbol{I}_{t,k}|\boldsymbol{I}_{t-1,k}, \mathcal{R}_{t,b}(\boldsymbol{S}_k^{(t-1)}), \lambda] \approx \phi(\boldsymbol{I}_{t,k}; \lambda \boldsymbol{X}_t \boldsymbol{z}_{t,k}^{w,b} + \alpha_t(1-d_t)\boldsymbol{I}_{t-1,k}, \|\lambda \boldsymbol{z}_{t,k}^{w,b}\|^2 \boldsymbol{\Sigma}_t), \tag{11}$$

where $\boldsymbol{z}_{t,k}^{w,b} = \mathcal{R}_{t,b}(\boldsymbol{S}_k^{(t-1)})\boldsymbol{Q}_t^w$ is an unscaled concentration vector and $\phi(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the probability density function of a multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. For each $b \in \{A, C, G, T\}$, we estimate $\lambda_{t,k}^b$ using the following optimization:

$$\lambda_{t,k}^b = \underset{\lambda}{\operatorname{argmax}}\, L_{t,k}^b(\lambda). \tag{12}$$

Our implementation of naiveBayesCall uses the golden section search method [10] to solve the 1-dimension optimization problem in (12).

Base-calling: The nucleotide at position t is called as

$$s_{t,k} = \underset{b \in \{A,C,G,T\}}{\operatorname{argmax}} \max_{\lambda} L_{t,k}^b(\lambda) = \underset{b \in \{A,C,G,T\}}{\operatorname{argmax}} L_{t,k}^b(\lambda_{t,k}^b), \tag{13}$$

and the sequence matrix is updated accordingly: $\boldsymbol{S}_k^{(t)} = \mathcal{R}_{t,s_{t,k}}(\boldsymbol{S}_k^{(t-1)})$.

Quality score: For position t, the probability of observing b is estimated by

$$\mathbb{P}(b) = \frac{\phi(\boldsymbol{I}_{t,k}; \lambda_{t,k}^b \boldsymbol{X}_t \boldsymbol{z}_{t,k}^{w,b} + \alpha_t(1-d_t)\boldsymbol{I}_{t-1,k}, \|\lambda_{t,k}^b \boldsymbol{z}_{t,k}^{w,b}\|^2 \boldsymbol{\Sigma}_t)}{\sum_{x \in \{A,C,G,T\}} \phi(\boldsymbol{I}_{t,k}; \lambda_{t,k}^x \boldsymbol{X}_t \boldsymbol{z}_{t,k}^{w,x} + \alpha_t(1-d_t)\boldsymbol{I}_{t-1,k}, \|\lambda_{t,k}^x \boldsymbol{z}_{t,k}^{w,x}\|^2 \boldsymbol{\Sigma}_t)}, \tag{14}$$

and the quality score for base b is given by

$$Q(b) = 10 \log_{10} \left[\frac{\mathbb{P}(b)}{1 - \mathbb{P}(b)} \right]. \tag{15}$$

4 Results

In this section, we compare the performance of our new algorithm naiveBayesCall with that of Bustard, Alta-Cyclic [5], and our original algorithm BayesCall [9].

4.1 Data and test setup

In our empirical study, we used a standard resequencing data of PhiX174 virus, provided to us by the DPGP Sequencing Lab at UC Davis. The data were obtained from a 76-cycle run on the Genome Analyzer II platform, with the viral sample in a single lane of the flow cell. The lane consisted of 100 tiles, containing a total of 14,820,478 clusters. Illumina's base-calling pipeline, called

Integrated Primary Analysis and Reporting, was applied to the image data to generate intensity files.

The entire intensity data were used to train Alta-Cyclic and BayesCall. Further, since Alta-Cyclic requires a labeled training set, the reads base-called by Bustard and the PhiX174 reference genome were also provided to Alta-Cyclic. To estimate parameters in BayesCall and naiveBayesCall, the intensity data for only 250 randomly chosen clusters were used.

To create a classification data set for testing the accuracy of the four base-calling algorithms, the sequences base-called by Bustard were aligned against the PhiX174 reference genome, and those reads containing more than 22 mismatches (i.e., with more than 30% of difference) were discarded. This filtering step reduced the total number of clusters to 6,855,280, and the true sequence associated with each cluster was assumed to be the 76-bp string in the reference genome onto which the alignment algorithm mapped the sequence base-called by Bustard. The same set of clusters was used to test the accuracy of all four methods.

Note that since the classification data set was created by dropping those clusters for which Bustard produced many errors, the above experiment setup slightly favored Bustard. Also, it should be pointed out that since Alta-Cyclic was trained on the entire lane, it actually had access to the entire testing data set during the training phase.

4.2 Improvement in running time

The experiments were done on a Mac Pro with two quad-core 3.0GHz Intel Xeon processors, utilizing all eight cores. Table 1(a) shows the training time and the prediction time of Alta-Cyclic, BayesCall, and naiveBayesCall. The times reported in Table 1(a) are for the full-lane of data. The training time of naiveBayesCall is the same as that of BayesCall, since naiveBayesCall currently uses the same parameter estimation method as in BayesCall. Although the training time of BayesCall is longer than that of Alta-Cyclic, we point out that, in principle, the cycle-dependent parameters in BayesCall can be estimated progressively as the sequencing machine runs (a run currently takes about 10 days). This advantage comes from the fact that BayesCall can be trained without labeled training data. As Table 1(a) illustrates, naiveBayesCall dramatically improves the base-calling time over BayesCall, delivering about 60X speedup. This improvement makes our model-based base-calling approach practical.

4.3 Summary of base-call accuracy

Table 1(b) shows the overall base-call accuracy of the four different methods. The columns under the label "4 Tiles" show the results for only 4 out of the 100 tiles in the lane. Since it would take more than 15 days for BayesCall to call bases for the entire lane, it was not used in the full-lane study. Both Bustard and Alta-Cyclic were trained on the full-lane data. To train BayesCall for the 4-tile data, we randomly chose 250 clusters from each tile to estimate tile-specific parameters,

Table 1. Comparison of overall performance results (a) Running times (in hours). (b) Base-call error rates. BayesCall's testing time was estimated from that for 4 tiles of data. The "by-base" error rate refers to the ratio of the number of miscalled bases to the total number of base-calls made, while the "by-read" error rate refers to the ratio of the number of reads each with at least one miscalled base to the total number of reads considered.

(a)

	Training Time	Testing Time for Full-Lane
Alta-Cyclic	10	4.4
BayesCall	19	362.5
naiveBayesCall	19	6

(b)

	4 Tiles		Full-Lane	
	By-base	By-read	By-base	By-read
Bustard	0.0098	0.2656	0.0103	0.2705
Alta-Cyclic	0.0097	0.3115	0.0101	0.3150
BayesCall	0.0076	0.2319	NA	NA
naiveBayesCall	0.0080	0.2348	0.0088	0.2499

and used the same parameters in naiveBayesCall. To run naiveBayesCall on the full-lane data, we randomly chose 250 clusters from the entire lane to estimate lane-wide parameters.

From Table 1(b), we see that the performance of naiveBayesCall is comparable to that of BayesCall. Figure 2 shows the tile-specific average error rate for each tile of the full-lane data. Note that naiveBayesCall clearly outperforms both Bustard and Alta-Cyclic for tiles 21 to 100, but has comparable error rates for tiles 1 to 20. It is possible to improve naiveBayesCall's accuracy for the first 20 tiles by using tile-specific parameter estimates (see Discussion).

Figure 3(a) illustrates the cycle-specific average error rate. Note that naiveBayesCall's average accuracy dominates Alta-Cyclic's for all cycles. Furthermore, the improvement of naiveBayesCall over Bustard increases with cycles, as illustrated in Figure 3(b). This suggests that it is possible to run the sequencing machine for longer cycles and still obtain useful sequence information for longer reads by using an improved base-calling algorithm such as ours. Furthermore, we believe that fewer errors in later cycles may facilitate de novo assembly. We return to this point in Section 4.5.

4.4 Discrimination ability of quality scores

To compare the utility of quality scores, we follow the idea in [6] and define the discrimination ability $D(\epsilon)$ at error tolerance ϵ as follows. First sort the called bases according to their quality scores in decreasing order. Then go down that sorted list until the error rate surpasses ϵ. The number of correctly called bases

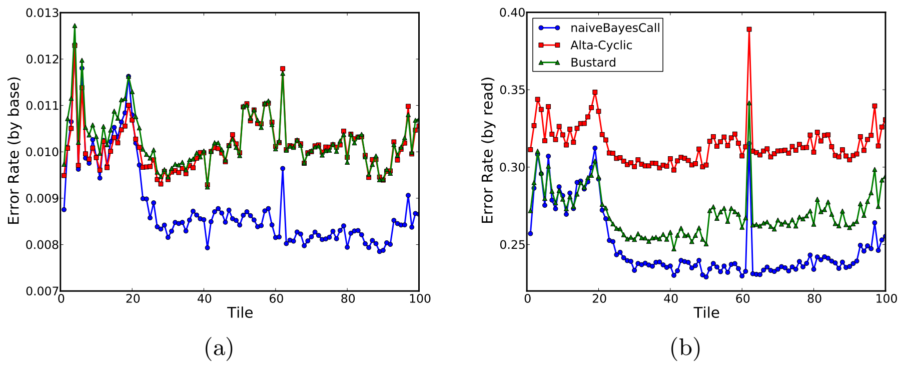

Fig. 2. Title-specific error rates. Our algorithm naiveBayesCall clearly outperforms both Bustard and Alta-Cyclic for tiles 21 to 100, but has comparable error rates for tiles 1 to 20. It is possible to improve naiveBayesCall's base-call accuracy for the first 20 tiles by using tile-specific parameter estimates. (a) By-base error rate for each tile. (b) By-read error rate for each tile.

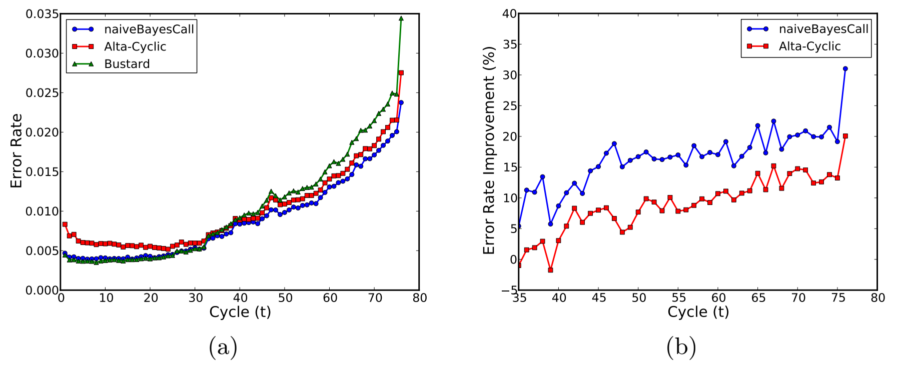

Fig. 3. Comparison of average base-call accuracy for the full-lane data. Note that naiveBayesCall's average accuracy dominates Alta-Cyclic's for all cycles. Further, The improvement of naiveBayesCall over Bustard increases with cycles. (a) Cycle-specific error rate. (b) Improvement of naiveBayesCall and Alta-Cyclic in cycle-specific error rate over Bustard.

up to this point is defined as $D(\epsilon)$. Hence, $D(\epsilon)$ corresponds to the number of bases that can be correctly called at error tolerance ϵ, if we use quality scores to discriminate bases with lower error probabilities from those with higher error probabilities. For any given ϵ, a good quality score should have a high $D(\epsilon)$. Shown in Figure 4 is a plot of $D(\epsilon)$ for naiveBayesCall, Alta-Cyclic, and Bustard. As the figure shows, naiveBayesCall's quality score consistently outperforms Alta-Cyclic's. For $\epsilon < 0.0017$ and $\epsilon > 0.0032$, naiveBayesCall's quality score has a higher discrimination ability than Bustard's, while the opposite is true for the intermediate values $0.0017 < \epsilon < 0.0032$.

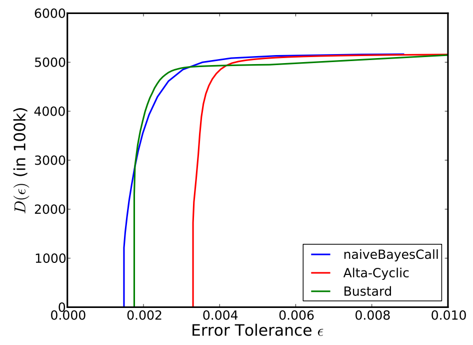

Fig. 4. Discrimination ability $D(\epsilon)$ of quality scores for the full-lane data.

4.5 Effect of base-calling accuracy on the performance of de novo assembly

Here, we demonstrate how improved base-calling accuracy may facilitate de novo assembly. Because of the short read length and high sequencing error rate, de novo assembly of the next-generation sequencing data is a challenging task. Recently, several promising algorithms [3, 4, 14, 17, 21] have been proposed to tackle this problem. In our study, we used the program Velvet [21] to perform de novo assembly of the reads called by different base-calling algorithms. First, we randomly chose a set of clusters from the 4-tile data without doing any filtering. Then, we base-called those clusters using each of Bustard, Alta-Cyclic, BayesCall, and naive-BayesCall, producing four different sets of base-calls on the same data set. For each set of base-called reads, Velvet was run with the k-mer length set to 55. For a given choice of coverage, we repeated this experiment 100 times. The results are summarized in Table 2, which shows the N50 length, the maximum contig length, and the total number of contigs produced; these numbers were averaged over the 100 experiments. On average naiveBayesCall led to better de novo assemblies than did Bustard or Alta-Cyclic: For 5X and 10X coverages, the performance of naiveBayesCall was similar to that of Bustard's in terms of the N50 and maximum contig lengths, but naiveBayesCall produced significantly more contigs than did Bustard. For 15X and 20X, naive-BayesCall clearly outperformed Bustard in all measures, producing longer and more contigs. The results for BayesCall and naiveBayesCall were comparable.

5 Discussion

Reducing the base-call error rate has important consequences for several subsequent computational problems, including assembly, SNP calling, disease association mapping, and population genomics analysis. In this paper, we have developed new algorithms to make our model-based base-calling approach scalable. Being a fully-parametric model, our approach is transparent and provides

Table 2. Average contig lengths resulting from de novo assembly of the 76-cycle PhiX174 data, when different base-calling algorithms are used to produce the input short-reads. The length of the PhiX174 genome is 5386 bp, and Velvet [21] was used to perform the assembly. N50 is a statistic commonly used to assess the quality of de novo assembly. It is computed by sorting all contigs by their size in decreasing order and adding the length of these contigs until the sum is greater than 50% of the total length of all contigs. The length of the last added contig is reported as N50. A larger N50 indicates a better assembly. Also shown are the maximum contig length (denoted "Max") and the total number of contigs (denoted "#Ctgs").

Cov-	Bustard			Alta-Cyclic			BayesCall			naiveBayesCall		
erage	N50	Max	#Ctgs	N50	Max	#Ctgs	N50	Max	#Ctgs	N50	Max	#Ctgs
5X	145	153	277	140	146	251	146	156	358	146	158	349
10X	203	368	2315	200	353	2148	203	368	2435	203	365	2467
15X	352	685	4119	331	637	4047	368	712	4249	371	716	4263
20X	675	1162	4941	674	1119	4893	752	1246	5004	750	1259	5015

quantitative insight into the underlying sequencing process. The improvement in base-call accuracy delivered by our algorithm implies that it is possible to obtain longer reads for a given error tolerance.

In our method, it is feasible to estimate parameters using a training set consisting of only a few hundred randomly chosen clusters. We believe that naiveBayesCall's ability to estimate local parameters using a small number of clusters should allow one to take into account the important differences between different tiles and lanes (recall the results discussed in Section 4.3). One possible approach to take in the future is to partition the lane into several regions and estimate region-specific parameters. Further, adopting the following strategy may work well: Estimate lane-wide parameters using a small number of clusters randomly chosen from the entire lane. Then, obtain tile-specific or region-specific parameter estimates by initializing with the lane-wide estimates and by performing a few iterations of the expectation-maximization algorithm (as described in [9]) using a small number of clusters from the tile or region. We believe that the accuracy of naiveBayesCall can be improved significantly by using tile-specific or region-specific parameter estimates.

Acknowledgments

We thank Kristian Stevens for useful discussion. This research is supported in part by an NSF CAREER Grant DBI-0846015, an Alfred P. Sloan Research Fellowship, and a Packard Fellowship for Science and Engineering.

References

1. D. R. Bentley. Whole-genome re-sequencing. *Curr. Opin. Genet. Dev.*, 16:545–552, 2006.

2. W. Brockman, P. Alvarez, S. Young, M. Garber, G. Giannoukos, W. L. Lee, C. Russ, E. S. Lander, C. Nusbaum, and D. B. Jaffe. Quality scores and SNP detection in sequencing-by-synthesis systems. *Genome Res.*, 18:763–770, 2008.
3. J. Butler, I. MacCallum, M. Kleber, I. A. Shlyakhter, M. K. Belmonte, E. S. Lander, C. Nusbaum, and D. B. Jaffe. ALLPATHS: De novo assembly of whole-genome shotgun microreads. *Genome Research*, 18(5):810–820, 2008.
4. M. J. P. Chaisson, D. Brinza, and P. A. Pevzner. De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome research*, 2008.
5. Y. Erlich, P. Mitra, M. Delabastide, W. McCombie, and G. Hannon. Alta-Cyclic: a self-optimizing base caller for next-generation sequencing. *Nat Methods*, 5:679–682, 2008.
6. B. Ewing and P. Green. Base-calling of automated sequencer traces using Phred. II. Error probabilities. *Genome Research*, 8(3):186–194, 1998.
7. I. Hellmann, Y. Mang, Z. Gu, P. Li, F. M. D. L. Vega, A. G. Clark, and R. Nielsen. Population genetic analysis of shotgun assemblies of genomic sequences from multiple individuals. *Genome Res*, 18(7):1020–9, 2008.
8. R. Jiang, S. Tavare, and P. Marjoram. Population genetic inference from resequencing data. *Genetics*, 181(1):187–97, 2009.
9. W. C. Kao, K. Stevens, and Y. S. Song. BayesCall: A model-based basecalling algorithm for high-throughput short-read sequencing. *Genome Research*, 19:1884–1895, 2009.
10. J. Kiefer. Sequential minimax search for a maximum. In *Proceedings of the American Mathematical Society*, volume 4, pages 502–506, 1953.
11. B. Langmead, C. Trapnell, M. Pop, and S. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3):R25, 2009.
12. H. Li, J. Ruan, and R. Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.*, 18:1851–8, 2008.
13. L. Li and T. Speed. An estimate of the crosstalk matrix in four-dye fluorescence-based DNA sequencing. *Electrophoresis*, 20:1433–1442, 1999.
14. P. Medvedev and M. Brudno. Ab Initio whole genome shotgun assembly with mated short reads. In *Proceedings of the 12th Annual Research in Computational Biology Conference (RECOMB)*, pages 50–64, 2008.
15. M. L. Metzker. Emerging technologies in DNA sequencing. *Genome Res*, 15(12):1767–76, 2005.
16. J. Rougemont, A. Amzallag, C. Iseli, L. Farinelli, I. Xenarios, and F. Naef. Probabilistic base calling of Solexa sequencing data. *BMC Bioinformatics*, 9:431, 2008.
17. A. Sundquist, M. Ronaghi, H. Tang, P. Pevzner, and S. Batzoglou. Whole-genome sequencing and assembly with high-throughput, short-read technologies. *PLoS One*, 2(5):e484, 2007.
18. A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, 1967.
19. N. Whiteford, T. Skelly, C. Curtis, M. Ritchie, A. Lohr, A. Zaranek, I. Abnizova, and C. Brown. Swift: Primary Data Analysis for the Illumina Solexa Sequencing Platform. *Bioinformatics*, 25(17):2194–2199, 2009.
20. Z. Yin, J. Severin, M. C. Giddings, W. A. Huang, M. S. Westphall, and L. M. Smith. Automatic matrix determination in four dye fluorescence-based DNA sequencing. *Electrophoresis*, 17:1143–1150, 1996.
21. D. R. Zerbino and E. Birney. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5):821–829, 2008.