A Novel Data Structure to Support Ultra-fast Taxonomic Classification of Metagenomic Sequences with k-mer Signatures
Supplementary materials

1 Proof of Theorem 1

Bipartite graph $G = (U, V, E)$ satisfies $|U| = m_a$, $|V| = m_b$, $E \subset U \times V$. Each edge $(u_i, v_j) \in E$ represents a $k$-mer $s$, $h_a(s) = i$ and $h_b(s) = j$. Since $h_a$ and $h_b$ are uniform random hash functions, these edges can be considered as randomly and uniformly chosen from all possible edges in $U \times V$ with probability $\frac{|E|}{|U||V|} = \frac{n}{m_am_b}$.

Consider a cycle in $G$, suppose the length of the cycle is $2t$. This cycle is equivalent to a list of $2t$ edges: $(u_1, v_1), (v_1, u_2), (u_2, v_2), \cdots, (u_t, v_t), (v_t, u_1)$. These $2t$ edges are uniquely decided by a list of $2t$ nodes $u_1, v_1, u_2, v_2, \cdots, u_t, v_t$. The number of such cycles that possibly exist in $G$ is

$$\binom{m_a}{t} \binom{m_b}{t}.$$ 

Here, $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. And each of these cycles exists with probability $(\frac{n}{m_am_b})^{2t}$.

Apply the conclusion presented in (Botelho et al., 2012) (Page 3), we know that when $\frac{n}{\sqrt{m_am_b}} < 1 - O\left(\frac{1}{\sqrt{m_am_b}}\right)$, which is always satisfied because $m_am_b \geq 1.33n^2$, the number of cycles with length $2t$ converges a Poission distribution with parameter $\lambda_t$, and

$$\lambda_t = \left(\frac{n}{m_am_b}\right)^{2t} \binom{m_a}{t} \binom{m_b}{t}.$$ 

Let $c = \frac{n}{\sqrt{m_am_b}}$, note that $t \ll m_a, t \ll m_b$, then we have

$$\lim_{t \to \infty} \frac{2t\lambda_t}{c^{2t}} = 1.$$ 

Hence the total number of cycles in $G$ converges a Possion distribution with parameter $\lambda$, where

$$\lambda = \sum_{t=1}^{\infty} \lambda_t = -\frac{1}{2} \ln(1-c^2)$$
2 Approaches of tuning $p_t$

$p(t)$ denotes the probability of an alien query returns $t$ for an $l$-Othello. Once $l$-Othello is constructed, the $p(t)$ values can be accordingly computed. In Othello, there are two array of $l$-bit integers, namely $A$ and $B$. We are able to modify the values in $A$ and $B$ without affecting any of the query results on the $l$-Othello. We describe two possible approaches as follows.

- Note that there are some elements in $A$ and $B$, these elements do not correspond to any $k$-mers. Hence, we can assign any $l$-bit integer value to each of them, so that the occurrence frequency of each element are balanced.

- For any connected component of the bipartite graph $G$, we can execute a XOR operation on all of its elements in $A$ and $B$. That is select any $l$-bit integer $x$ and replace all $A[i]$ values in the connected component by $A[i] \oplus x$ and replace all $B[j]$ values by $B[j] \oplus x$ simultaneously. As long as for each connected components of $G$ the corresponding values are all simultaneously replaced, this operation does not affect any $\tau(s)$ values.

In practice, we can always tune the values so that $p(t)$ is of the same order of magnitude for all $t$, and all of them are approximately $2^{-l}$.

3 Proof of Theorem 2

We analyze the confidence of a $K$-mer window as follows. For a window of $k$-mers, let $w$ be the length of the window. Suppose the query result for these $K$-mers are $\tau(s_1), \tau(s_2), \cdots, \tau(s_w)$. For a particular level of the taxonomy tree, suppose that these $k$-mer belongs to taxon $t$, then $\tau(s_1), \tau(s_2), \cdots, \tau(s_w) \in S_t$, where $S_t$ is the set of the IDs of the nodes in the taxonomy subtree with the root $t$.

For consecutive $w$ $k$-mers, let $G_t$ be the event that this window is actually from taxon $t$. We assume there is no sequencing error, hence, when $G_t$ the query results for these $w$ $k$-mers satisfy $\tau(s_1), \tau(s_2), \cdots, \tau(s_w) \in S_t$. We use notation $Q_t$ to describe the event that $\tau(s_1), \tau(s_2), \cdots, \tau(s_w) \in S_t$.

Let $G_t$ be the event that this window is actually from taxon $t$. We assume there is no sequencing error, hence, when $G_t$ the query results for these $w$ $k$-mers satisfy $\tau(s_1), \tau(s_2), \cdots, \tau(s_w) \in S_t$. We use notation $Q_t$ to describe the event that $\tau(s_1), \tau(s_2), \cdots, \tau(s_w) \in S_t$.

Now the problem is that if we observe event $Q_t$, we may indicate two reasons exclusively. (1) $Q_t$ happens as a result of $G_t$. (2) Note that for alien $k$-mers $\tau$ may return any integer, $Q_t$ happens as a result of the query result of $w$ alien $k$-mers. We use the...
probability \( P(G_t|Q_t) \) to describe how confident we are, about that this window is from taxon \( t \).

As described, when \( G_t \) happens, \( Q_t \) also happens. Hence \( P(Q_t|G_t) = 1 \).

We estimate the value of \( P(G_t|Q_t) \) as follow.

\[
P(G_t|Q_t) = \frac{P(Q_t|G_t)P(G_t)}{P(Q_t|G_t)P(G_t) + P(Q_t|\overline{G_t})P(\overline{G_t})} = \frac{P(G_t)}{P(G_t) + P(Q_t|\overline{G_t})P(\overline{G_t})} \tag{1}
\]

Let \( q_t \) be the abundance of the window from taxon \( t \), i.e., for a particular sample, randomly select one window of length \( w \) among all windows in all reads from this sample, the probability that this window is actually from taxon \( t \). Hence \( P(G_t) = q_t \).

The value \( P(Q_t|G_t) \) is estimated as follow.

\( \overline{G_t} \) means that this window is not from taxon \( t \). \( \overline{G_t} \) indicates either one of the following sub-events: (1) \( C_{\text{other}} \): In this particular level of taxonomy tree, the window is from one other taxon \( t' \), which means the query results \( \tau(s_1), \tau(s_2), \ldots, \tau(s_w) \in S_{t'} \) for a \( t' \neq t \). Note that \( S_{t'} \cap S_t = \emptyset \), this indicates \( P(Q_t|C_{\text{other}}) = 0 \). (2) \( C_{\text{alien}} \): This window is an alien of the taxonomy tree. Let \( c_t = P(C_{\text{alien}}|\overline{G_t}) \), then \( 0 < c_t < 1 \).

\[
P(Q_t|G_t) = P(Q_t|C_{\text{other}})P(C_{\text{other}}|\overline{G_t}) + P(Q_t|C_{\text{alien}})P(C_{\text{alien}}|\overline{G_t}) = P(Q_t|C_{\text{alien}})c_t \tag{2}
\]

As discussed in Section 2.2.3,

\[
P(Q_t|C_{\text{alien}}) = q(t)^w \tag{3}
\]

Combine Equation (1) (2) (3), we have

\[
P(G_t|Q_t) = \frac{q_t}{q_t + (p(t))^w c_t} \tag{4}
\]

Note that, \( q_t > 0 \) and \( 0 < p(t) \ll 1 \). Hence \( P(G_t|Q_t) \to 1 \) as \( t \to \infty \). This is to say when \( w \) increases, \( P(G_t|Q_t) \) also grows, and we can be more confident that when a query result shows that a window belongs to some taxon \( t \), it reflects the fact that this window is actually from this taxon \( t \). In other words, a longer window is more likely to come from this taxon than a shorter one.

Note that

\[
\frac{q_t}{q_t + (p(t))^w c_t} \gg \frac{q_t}{q_t + (p(t))^w}
\]

We use a threshold value \( \lambda \), when \( P(G_t|Q_t) > 1 - \lambda \), we accept, which is equivalent to:

\[
w > \log_{p(t)} \left( \frac{\lambda q_t}{1 - \lambda} \right) \sim \log_{p(t)} \lambda q_t
\]
Here, the value of $q_t$ can not be directly measured. However, for any actually detected taxon, we are sure that $q_t \geq \frac{1}{M}$, where $M$ is the total number of reads in the dataset. Hence we use the following threshold to decide the length of accepted windows.

$$w > \log_{p(t)} \left( \frac{\lambda}{(1-\lambda)M} \right) \sim \log_{p(t)} \left( \frac{\lambda}{M} \right)$$

Note that we can always use the $l$-Othello to compute the value of $p(t)$. Thus, given $\lambda$ ($\lambda = 0.001$ by default), for each taxon, we can pre-compute the minimum size threshold for $K$-mer window. Only the $K$-mer windows which are not shorter than its associated minimum window size will be accepted for final assignment determination.

4 Implementation of MetaOthello

Jellyfish is used to collect all distinct $k$-mers from the designated reference genome database. For each $K$-mer, we counted its frequencies among all taxa at each taxonomic rank and also stores the first taxon it appears in. And a $K$-mer will be assigned to a taxon-specific $K$-mer set if it exists and only exists in that taxon for that taxonomic level, and its frequency is larger than 1 at the next level.

$l$-Othello will be built given the set of $k$-mers and their associated taxon IDs.

During the classification of the sequencing reads, $l$-Othello will be loaded into the memory first. Read will be classified one at a time sequentially. In the case of paired-end reads, information from both ends will be combined as one score when selecting the best assignment.

5 Results of count-based MetaOthello

To investigate how the window-based approach helps MetaOthello in sequencing read classification, we implemented and ran a count-based version MetaOthello on the sequencing datasets used in section 3.1 and 3.2. Figure 1 shows the correlation of species-specific $k$-mer signatures with classification accuracy for both window-based MetaOthello and count-based MetaOthello. Clearly, using both 20-mers and 31-mers, the window-based implementation exhibits higher accuracy. Table 1 presents the results (read assignment precision, sensitivity, and F-score) of count-based MetaOthello. Compared with the results of the default window-based MetaOthello in section 3.2, significant decreases on precision can be easily found for count-based MetaOthello.
Figure 1: Correlation of species-specific $k$-mer proportion with classification accuracy for window-based MetaOthello and count-based MetaOthello when $k=20$ (A) and $k=30$ (B).
Table 1: Count-based MetaOthello read assignment precision, sensitivity, and F-score.

<table>
<thead>
<tr>
<th>Phylum</th>
<th>Genus</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec / Sens / F-score</td>
<td>Prec / Sens / F-score</td>
</tr>
<tr>
<td>HiSeq</td>
<td>20mer</td>
<td>95.7 / 95.1 / .954</td>
</tr>
<tr>
<td></td>
<td>25mer</td>
<td>94.8 / 93.7 / .943</td>
</tr>
<tr>
<td></td>
<td>31mer</td>
<td>93.7 / 91.8 / .927</td>
</tr>
<tr>
<td>MiSeq</td>
<td>20mer</td>
<td>98.3 / 97.8 / .980</td>
</tr>
<tr>
<td></td>
<td>25mer</td>
<td>98.1 / 95.3 / .967</td>
</tr>
<tr>
<td></td>
<td>31mer</td>
<td>97.9 / 92.6 / .951</td>
</tr>
<tr>
<td>SimBA5</td>
<td>20mer</td>
<td>98.6 / 98.6 / .986</td>
</tr>
<tr>
<td></td>
<td>25mer</td>
<td>97.5 / 97.4 / .976</td>
</tr>
<tr>
<td></td>
<td>31mer</td>
<td>93.9 / 93.0 / .935</td>
</tr>
</tbody>
</table>

Table 2: Kaiju read assignment precision, sensitivity, and F-score using the indices built on the two other options of source databases.

6 Results of Kaiju using the indices built on the two other options of source databases

In section 3.2, to conduct the comparative studies in a fair manner, we ran Kaiju using the same source database as the other three tools (MetaOthello, Kraken, and Clark). We notice that in Kaiju's manual (https://github.com/bioinformatics-centre/kaiju/blob/master/README.md), there two additional recommended databases (nr and proGenomes). To investigate how the choice of source database affects its performance, we further ran Kaiju using both of the two databases. As reported in Table 2, though some improvements are achieved (sensitivities at the phylum/genus level on HiSeq/MiSeq data), its performance is still far behind that of MetaOthello, Kraken, and Clark.
References


