Optimising Locality-Sensitive Hashing on Sequences in the Context of Motif Finding

Warren A. Cheung[a*]

[a]Bioinformatics Program, Centre for Molecular Medicine and Therapeutics
University of British Columbia, Vancouver, BC

ABSTRACT

Locality-sensitive hashing has been applied in several problems in bioinformatics to quickly search large sequences by examining the n-grams of the sequences. We observe in these application a bias in the random generation of hashes and define an effective equivalence for hashes that generate nearly identical results. Methods that address these two points demonstrate an improvement to the rate at which unique hashes are generated, especially when many hashes are generated. These methods can be easily integrated with existing applications of locality-sensitive hashing, resulting in improved performance by reducing the time algorithms spend revisiting duplicate hashes and eliminating search biases.

Keywords: locality-sensitive hashing, projection, motif-finding, n-grams, pattern discovery, similarity search

1 Introduction

The random projection method of locality sensitive hashing is a stochastic method of dimensionality reduction. This technique has been successfully applied to many problems involving identification of patterns, such as motif-finding and local sequence alignment. This paper discusses optimisations in the context of the motif finding problem originally proposed by Pevzner and Sze[1], with the application of locality sensitive hashing method PROJECTION described by Buhler and Tompa[2], which uses random projection as a efficient method for finding common semi-conserved motifs of a set length without gaps from a set of sequences. We demonstrate that careful generation around a set of constraints can result in a significantly reduced search space with effectively equivalent results.

Previously, Raphael et al. developed UNIFORM PROJECTION[3], another improvement of PROJECTION that samples the projection space in a more uniform manner. We note that the optimizations discussed here are orthogonal from those, and further improvement to performance may be possible by combining with their work, such as previously achieved by Wang and Yang in UPNT[4] combining uniform projection with unpublished work on neighbourhood thresholding[5]. As well, the same analysis generalises to other locality-sensitive hashing-based methods, especially those involved in *n*-gram analysis, such as large-scale sequence comparison[6].

1.1 Classic Random Projection Algorithm

In the motif finding problem originally described by Pevzner and Sze[1], let S be a set of t input sequences with length n, and a motif of known length l with at most d mutations be planted in each of the t sequences. The purpose of the projection algorithm is to extract the t planted instances of the motif from S. For each sequence S_i in S, a sliding window of length l is run over S_i to generate a set of $n-l+1$ "l-mers" for S_i. Characters in a string shall be referred with 0 as the first character of the string. Let $s_{i,j}$ refer to the j^{th} symbol in S_i, where $0 \leq j < n$. Likewise, let $L_{i,j}$ refer the l-mer from S_i starting with symbol $s_{i,j}$, such that $L_{i,j} = s_{i,j}s_{i,j+1}...s_{i,j+l-1}$. Let L be the set of all l-mers.

A projection P is a set of k positions, each position representing one of the l positions $\{0,1,\ldots,l-1\}$. In random projection, each position is chosen by selecting one of the l positions uniformly at random with replacement and adding the position selected to P. This process is repeated k times. As the positions are chosen with replacement, a projection can specify from 1 to k distinct positions.

Given a projection P, each of the l-mers from S are hashed into a "bucket" by the bases of the l-mer at the positions specified by P. For a projection P of positions $\{p_1, p_2, \ldots, p_k\}$ and l-mer $L_{i,j}$, let $P(L_{i,j}) = s_{i,j+p_1}s_{i,j+p_2}\ldots s_{i,j+p_k}$. A bucket $B_{P,x}$ is the set of all l-mers $L_{i,j}$ where $P(L_{i,j}) = x$. A bucket containing an unusually large number of l-mers has an elevated chance of being enriched for a motif and is used as the basis for a "refinement" step to extract a motif

*E-mail: wcheung@cmmt.ubc.ca, Telephone: +1(604)875-2345 ext. 7947.

via an EM-based approach (see Buhler and Tompa[2] for more details). We shall demonstrate that the method of choosing positions uniformly at random with replacement results in biased selection of projections.

2 Selection Bias

The current selection method introduces a bias in the selection of potential projections towards those that can be generated in multiple ways if the positions are selected uniformly at random with replacement. Naively, selecting from n possible positions k times results in n^k possible projections. However, it is obvious that the order of positions chosen for projection does not matter. Take two projections P and Q which are identical other than the order of the projected positions. Let f be a function that reorders the positions in P to the positions of Q. We can then note that the number of l-mers in bucket $B_{P,x}$ is the same as bucket $B_{Q,f(x)}$.

Definition Projection P is *equivalent* to Q if there exists a correspondence f such that for any set of sequences, for every l-mer $L_{i,j}$, $f(P(L_{i,j})) = Q(L_{i,j})$. P and Q are *exactly equivalent* if P is equivalent to Q and vice versa.

Definition Projection Q is *non-redundant* if for $Q = \{q_1, q_2, \dots, q_k\}$, for any q_i, q_j where $i \neq j$, $q_i \neq q_j$.

Under this definition of equivalence, it can be shown that any projection P is exactly equivalent to a non-redundant projection $Q - f$ in this case reorders the positions as needed with duplicate positions removed. Therefore, we shall use the following *standard form* for listing positions in projections: (i) the projected positions are in sorted order and (ii) no positions are duplicated. Note that all projections are equivalent to a projection in standard form, and no two different projections in standard form are equivalent. If only projections in standard form are considered, the search space of possible projections is reduced to $\sum_{i=1}^{k} \binom{n}{i}$.

When selecting positions uniformly at random with replacement, selection bias occurs because some projections can be generated by more combinations of randomly selected positions than others. For example, for $k = 3$, the only way to randomly generate the projection $\{5\}$ is by selecting position 5 for all three random draws $(5,5,5)$. This yields the non-redundant projection $\{5\}$, and no other projection can be generated that is equivalent to this projection. However, the draws $(3,1,3)$, $(1,3,3)$, $(3,3,1)$, $(3,1,1)$, $(1,3,1)$ and $(1,1,3)$ all generate a non-redundant projection of $\{1,3\}$, making it three times as likely to be generated as $\{5\}$.

The selection bias increases duplicate projections. As the projection algorithm is deterministic, duplicate projections cause redundant computation of already-evaluated results. Even when duplicate projections are not chosen, the projection choices are biased towards projections that have more ways to be generated. Such a bias makes the method less effective when the solution involves a less likely to be generated projection.

2.1 Avoiding Selection Bias

An efficient method to generate projections with equal frequency is to modify the random selection process. The new process selects k elements at random without replacement from the set $\{0, 1, \dots, l-1\} \cup \{\emptyset_0, \emptyset_1, \dots \emptyset_{k-2}\}$. The $k-1$ elements $\emptyset_i$ are *pseudo-positions* not in the set of original positions $\{0, 1, \dots, l-1\}$. Let S be the set of k selected elements. If S contains no pseudo-positions or $S \cap \{\emptyset_0, \emptyset_1, \dots \emptyset_{k-2}\} = \{\emptyset_0, \emptyset_1, \dots \emptyset_i\}$ for some i, S defines the projection $S \cap \{0, 1, \dots l-1\}$. Otherwise, S is discarded and the selection process is restarted from the beginning.

This method is slightly more costly than sampling at random, as several restarts may be necessary before a valid projection is generated. However, it guarantees every potential projection is generated with equal frequency, as there a set of equivalent projections can only be generated in one manner.

3 Effective Equivalence of Projections

As the l-mers in projection are generated by sliding a window across the input sequences, sequence positions are always shared between neighbouring l-mers. A position $s_{i,j}$ appears in up to l of the l-mers: $L_{i,j-l+1}, L_{i,j-l+2}, \dots, L_{i,j}$. One side effect of this effect is that many combinations of projections P and l-mer $L_{i,j}$ will result in the same sequence $P(L_{i,j})$. For example, let projections P and Q be $P = \{2,4\}$ and $Q = \{1,3\}$. For $1 \leq j \leq n-1$, $P(L_{i,j}) = Q(L_{i+1,j+1})$. This motivates a more relaxed definition of equivalence between two projections.

Definition A mapping Δ from $L' \to L$ is a *shift* of L if $\Delta(L_{i,j}) = L_{i,j-\delta}$ and $\delta \geq 0$, where L' is the set of all l-mers $L_{i,j} \in L, j \geq \delta$.

The shift Δ maps nearly all the l-mers by a constant shift along the sequence to a nearby l-mer. However, a constant number of l-mers at the start of each sequence are unable to be mapped.

Definition Projection P is *effectively equivalent* to Q if there exists a shift Δ such that for any set of sequences, for every l-mer mapped by Δ , $P(\Delta(L_{i,j})) = Q(L_{i,j})$.

This relaxes our definition of equivalence, allowing P and Q to be equivalent as long as for the l-mers mapped by Δ (all l-mers except a small constant number δ), a projection $Q(L_{i,j})$ can be mapped to the projection $P(\Delta(L_{i,j}))$. Note when we have a shift $\delta = 0$, we have exact equivalence, therefore effective equivalence includes exact equivalence.

One specific case of such an effective equivalence is that when $0 \notin P$, P is effectively equivalent to the projection Q where $Q = \{x | (x + \min(P)) \in P\}$. This case corresponds to a shift Δ where $\delta = \min(P)$.

Proof. By contradiction, assume there exists a projection P, $0 \notin P$, where P is not effectively equivalent to Q, where $q_i = p_i - \min(P)$ and $\Delta(L_{i,j}) = L_{i,j-\min(P)}$. Then there must exist an l-mer $L_{i,j}$ such that $P(\Delta(L_{i,j})) \neq Q(L_{i,j})$. However,

$$\begin{aligned} P(\Delta(L_{i,j}))) &= P(L_{i,j-\min(P)}) \\ &= s_{i,j+p_1-\min(P)} s_{i,j+p_2-\min(P)} \cdots s_{i,j+p_k-1-\min(P)} \\ &= s_{i,j+q_1} s_{i,j+q_2} \ldots s_{i,j+q_k-1} = Q(L_{i,j}). \end{aligned}$$

Therefore, no such l-mer $L_{i,j}$ exists and therefore P is effectively equivalent to Q. □

Therefore, any projection P is effectively equivalent to a projection Q in standard form, where $\min(Q) = 0$. This reduces the space of possible projections where no two projections are effectively equivalent to at most $\sum_{i=1}^{k-1} \binom{n}{i}$.

Note that the generation of effectively equivalent projections is also a source of bias in the generation of projections, as some projections are effectively equivalent to more projections than others. For example, when $l = 3$, the projection $\{3,4,6\}$ is also effectively equivalent to the projections $\{2,3,5\}$, $\{1,2,4\}$ and $\{0,1,3\}$ whereas the projection $\{0,2,6\}$ has no other effectively equivalent projection and so is four times less likely to be generated (assuming the unbiased generation of projections described in Section 2.1).

3.1 Avoiding Effectively Equivalent Projections

To avoid generating effectively equivalent projections, it is simply necessary to ensure that the position 0 is part of the projection generated. The simplest solution is what we shall refer to as 0-base Projections, which is to ensure that the first position selected for any projection is position 0. The remaining positions can be selected as before, with the only other change being that position 0 be excluded from the future draws (as it has already been generated).

4 Results

To demonstrate the effect of these optimisations, the number of unique projections, or hashes, generated using the original basic hashing method (as seen in UPNT[4]) was compared to the number of unique hashes generated by the unbiased hashing method (See Section 2.1). Experiments were performed 100 times for each condition. The one-tailed unpaired t-test assuming non-equal variance was used to compute p-values. The test condition was projecting up to $k = 7$ positions of $l = 15$ l-mers. This test condition replicates the parameters for PROJECTION when solving the classic $(15-4)$ $(l-d)$ motif-finding problem[2]. Timing results used a Python (`http://www.python.org`) implementation[1] and the internal `timeit.Timer` class, running the hashing function 1000 times.

4.1 Uniqueness of Hash Generation Methods

Figure 1 compares the number of unique hashes – hashes that are not equivalent. The unbiased hash function generates more unique hashes than the basic hash function, markedly so as the number of iterations increases. However, the difference is significant even at the smallest number of iterations tested ($p < 1.24 \times 10^{-7}$ at 500 iterations). A five percent improvement in the number of unique hashes generated is seen by 7500 iterations.

Even more striking is the number of hashes which are effectively unique – hashes which are not effectively equivalent. As the number of iterations increases, many of the hashes generated are effectively equivalent to a previously generated hash. At the extreme, after 25000 iterations, an average of 12819 unique hashes were generated by the unbiased hash generation method, but only 5966 of these were effectively unique – not effectively equivalent to a previously generated hash. To compare, 11659 unique hashes were generated on average by the original hash generation method, of which 5428 were effectively unique. The difference in effectively unique hashes generated is significant even at the smallest number of iterations tested ($p < 3.4 \times 10^{-23}$ at 500 iterations). Five percent improvement was seen by 2000 iterations (See Figure 3).

Figure 2 shows the effect of using the 0-base method on generating effectively unique hashes. The number of effectively equivalent hashes is increased substantially as effectively equivalent hashes are no longer generated. 0-base hash generation dominates the original hashing methods. As well, using 0-base, unbiased hash generation generates more effectively unique hashes than than basic hash generation, even after the smallest number of iterations tested ($p < 7.06 \times 10^{-11}$ at 500 iterations). Using 0-base unbiased hashing has over five percent improvement over basic hashing at 500 iterations, rising to 26% by 10000 iterations (See Figure 3).

[1]Source and raw data at `http://dnahelix.wikidot.com/locality-sensitive-hashing-for-motif-finding`

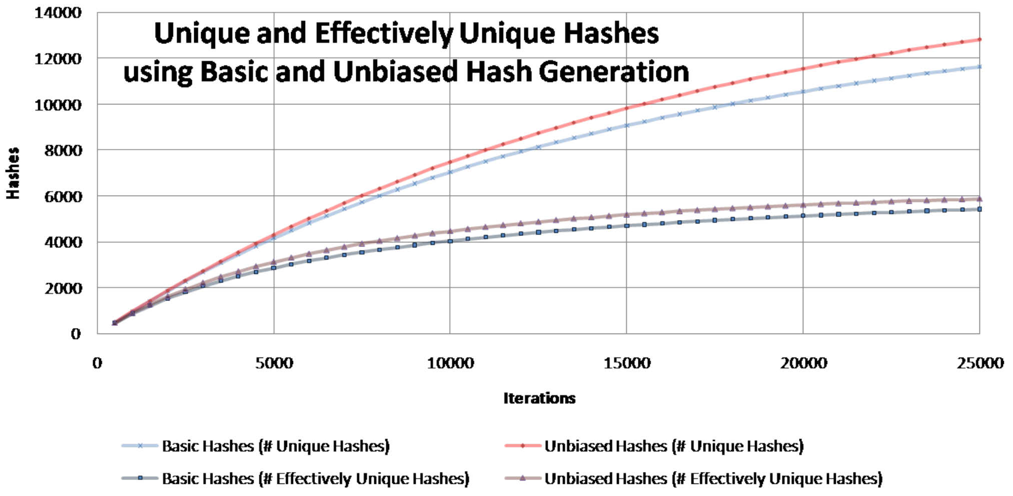

Figure 1. Average number of hashes that are unique and effectively unique over 100 trials. Unbiased hashing produces more hashes than basic hashing, however, the number of unique hashes is greatly reduced if we consider effective equivalence.

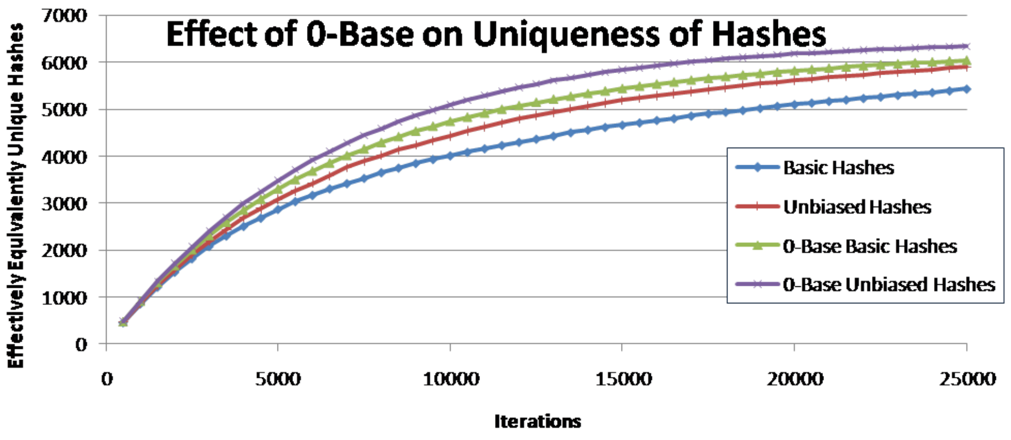

Figure 2. Average number of effectively unique hashes over 100 trials using unbiased and/or 0-base hashing. The unbiased hash generates more effectively unique hashes than the basic hash, and using 0-base increases this further.

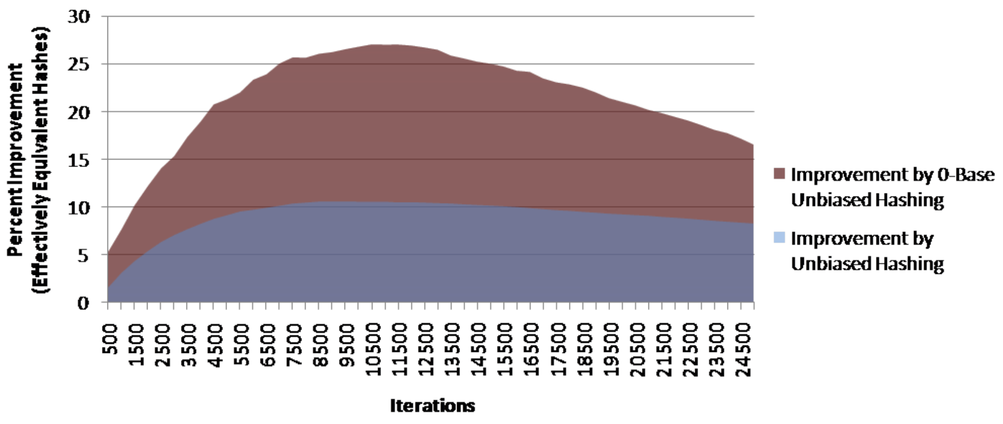

Figure 3. Precentage improvement on the number of effectively unique hashes, comparing unbiased hashing and unbiased with 0-base hashing over basic hashing. Using both types improvements, over 27% improvement was seen after 10500 iterations.

4.2 Runtime

The main caveat to using unbiased hash generation is the performance penalty incurred when rejecting hashes that are not in the desired format, which can happen with significant frequency. For our test case, the unbiased hash generation took on average 7.00×10^{-5}s to execute, whereas the basic hash generation took only 5.45×10^{-6}s. However, this is mitigated by the fact that selecting a hash function is a relatively short step in the motif finding algorithm, and would result in a net gain in cases where significantly more computation time is spent applying the hash function and analysing the result, as is generally the case. One refinement would be the pseudo-random generation of the hashes, which could guarantee no duplicates are generated. The application of uniform projection[3] or storing and rejecting all previously generated hashes could also mitigate this effect.

On the other hand, avoiding effectively equivalent projections actually simplifies generating the hash, as there is one fewer random trial to perform. The unbiased hash generation took slightly less time than before, on average 6.12×10^{-5}s. The basic hash generation was also only negligibly improved to 5.39×10^{-6}s.

5 Conclusion

We have demonstrated here two methods to reduce the search space of possible hashes for applications of locality-sensitive hashing in motif discovery. We note a bias in the random generation of hashes and define an effective equivalence for hashes that generate nearly identical results. Methods to address both these points are demonstrated to improve the uniqueness of hashes generated, especially when many hashes are generated.

Acknowledgements

This work was made possible due to funding by the CIHR/MSFHR Strategic Training Program in Bioinformatics, Natural Sciences and Engineering Research Council of Canada, the Michael Smith Foundation for Health Research, the Ontario Institute for Cancer Research and the University of British Columbia. Thanks to William S. Evans and Leon French for their input.

References

[1] P. A. Pevzner and S.-H. Sze, “Combinatorial approaches to finding subtle signals in DNA sequences,” in *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB-00)*, pp. 269–278, AAAI Press, (Menlo Park, CA), Aug. 16–23 2000.

[2] J. Buhler and M. Tompa, “Finding motifs using random projections,” *Journal of Computational Biology* **9**, pp. 225–242, Apr. 2002.

[3] B. Raphael, L.-T. Liu, and G. Varghese, “A uniform projection method for motif discovery in DNA sequences,” *IEEE/ACM Trans. Comput. Biol. Bioinformatics* **1**(2), pp. 91–94, 2004.

[4] J. Wang and D. Yang, “UPNT: Uniform projection and neighbourhood thresholding method for motif discovery,” *International Journal of Bioinformatics Research and Applications* **4**(1), pp. 96–106, 2008.

[5] J. King, W. Cheung, and H. Hoos, “Neighbourhood thresholding for projection-based motif finding.” Unpublished Manuscript.

[6] J. Buhler, “Efficient large-scale sequence comparison by locality-sensitive hashing,” *Bioinformatics* **17**(5), pp. 419–428, 2001.