Fast Calculation of Pairwise Mutual Information for Gene Regulatory Network Reconstruction

Peng Qiu, Andrew J. Gentles and Sylvia K. Plevritis

Department of Radiology, Stanford University, Stanford, CA

Abstract

We present a new software implementation to more efficiently compute the mutual information for all pairs of genes from gene expression microarrays. Computation of the mutual information is a necessary first step in various information theoretic approaches for reconstructing gene regulatory networks from microarray data. When the mutual information is estimated by kernel methods, computing the pairwise mutual information is quite time-consuming. Our implementation significantly reduces the computation time. For an example data set of 336 samples consisting of normal and malignant B-cells, with 9563 genes measured per sample, the current available software for ARACNE requires 142 hours to compute the mutual information for all gene pairs, whereas our algorithm requires 1.6 hours. The increased efficiency of our algorithm improves the feasibility of applying mutual information based approaches for reconstructing large regulatory networks.

Keywords: mutual information, gene regulatory network, microarray.

1 Introduction

Information theoretic approaches are increasingly being used for reconstructing regulatory networks from gene expression microarray data. Examples include the Chow-Liu tree [1], the relevance network (RelNet) [2], the context likelihood of relatedness (CLR) algorithm [3], ARACNE [4] and the maximum relevance minimum redundance network (MRNet) [5]. These specific approaches and others start by computing the pairwise mutual information (MI) for all possible pairs of genes, resulting in an MI matrix. The MI matrix is then manipulated to identify regulatory relationships. The Chow-Liu tree is a maximum spanning tree constructed from the MI matrix, where edges are weighted by the MI values between connected nodes. In relevance networks, an edge exists between a pair of genes if their MI exceeds a given threshold. The CLR algorithm transforms the MI matrix into scores that take into account the empirical distribution of the MI values, and then applies a threshold. ARACNE applies the data processing inequality (DPI) on each connected gene triple, removing potential false positive edges in the relevance networks. In MRNet, the maximum relevance minimum redundance (MRMR) criterion [6] is applied, where the maximum relevance criterion tends to assign edges to gene pairs that share large MI, while the minimum redundance criterion controls false positives. The MRMR criterion is essentially a pairwise approximation of the conditional MI. In [7], the conditional MI is explicitly used for inference. If two genes share large MI but are conditionally independent give a third gene, there is no edge between them. These approaches have been applied successfully to simulated data and real microarray data, identifying interesting regulatory targets and pathways.

The most time-consuming step in these approaches is computing the MI matrix, because all possible pairs of genes need to be examined. When the expression values of genes are treated as continuous random variables and the MI is estimated by kernel methods, computing the pairwise MI can be computationally expensive. For example, in a recently analyzed B-cell data set consisting of 336 samples and 9563 genes per sample, [4], ARACNE takes 142 hours to compute the MI for all gene pairs.

We propose a faster way to calculate the pairwise MI, where the order of the calculations is rearranged so that repeating operations are significantly reduced. For the previously mentioned B-cell data set, the run time of our algorithm is 1.6 hours. To explain our algorithm, in Section 2, we first briefly review kernel based estimation of MI. Then we present the pseudo-code of our approach. Section 3 illustrates the decrease in computation time, and is followed by Conclusions.

2 Method

2.1 Kernel Estimation of Mutual Information

The MI between two continuous random variables X (a gene) and Y (another gene) is defined as follows:

$$I(X;Y) = \iint f(x,y) log \left(\frac{f(x,y)}{f(x)f(y)} \right) dxdy \tag{1}$$

where $f(x,y)$ is the joint probability density of the two random variables, and $f(x)$ and $f(y)$ are the marginal densities. Given M data points drawn from the joint probability distribution, (x_j, y_j), $j = 1, ..., M$, the joint and marginal densities can be estimated by the Gaussian kernel estimator [8], where

$$f(x,y) = \frac{1}{M} \sum \frac{1}{2\pi h^2} e^{-\frac{1}{2h^2}((x-x_j)^2+(y-y_j)^2)} \tag{2}$$

$$f(x) = \frac{1}{M} \sum \frac{1}{\sqrt{2\pi h^2}} e^{-\frac{1}{2h^2}(x-x_j)^2} \tag{3}$$

and $f(y)$ takes a similar form. h is a tuning parameter that controls the width of the kernels. Since MI is an integration with respect to the joint probability density, it can be written as an expectation with respect to the random variables X and Y, and thus can be approximated by the sample average,

$$I(X;Y) = \frac{1}{M} \sum_i log \frac{M \sum_j e^{-\frac{1}{2h^2}((x_i-x_j)^2+(y_i-y_j)^2)}}{\sum_j e^{-\frac{1}{2h^2}(x_i-x_j)^2} \sum_j e^{-\frac{1}{2h^2}(y_i-y_j)^2}} \tag{4}$$

2.2 Existing Method for Pairwise Mutual Information

When the pairwise MI among N random variables (N genes) is considered, Equation (4) can be calculated inside two nested for-loops, both of which run from 1 to N. The software implementation for ARACNE [4] and MRNet [5] both adopt this structure, as did our initial implementation. However, we found this structure to be inefficient when coupled with kernel density estimator, because of repeated calculation of kernel distances.

When estimating the MI between two random variables g_1 and g_2, Equation (4) evaluates the kernel distances between all possible pairs of data points for each of the two random variables separately, $e^{-\frac{1}{2h^2}(g_{1_i}-g_{1_j})^2}$ and $e^{-\frac{1}{2h^2}(g_{2_i}-g_{2_j})^2}$, for all $i, j = 1, 2, ..., M$. When estimating the MI between two random variables g_1 and g_3, the kernel distances $e^{-\frac{1}{2h^2}(g_{1_i}-g_{1_j})^2}$ are evaluated again. Therefore, when Equation (4) is computed inside two nested for-loops, the kernel distances among data points of one random variable are evaluated $N-1$ times. For the large N in gene expression microarray data, the repeated calculations will severely decrease the computational efficiency. Avoiding the repeated calculations can significantly improve the computational efficiency of ARACNE. (Note: The implementation of MRNet in the Bioconductor package does not suffer from this problem, because it does not use kernel density estimation.)

A straight forward method to eliminate the repeats would be to pre-compute the kernel distances for each random variable (gene). However, such a method is not feasible because a large amount of memory would be needed to store the kernel distances, which is a matrix of size $N \times M^2$. In the following section, we present an alternative fast algorithm for calculating pairwise MI, where repeats are significantly decreased, with a moderate increase in memory use.

2.3 Fast Calculation of Pairwise Mutual Information

In practice, when we encounter computational structures such as nested for-loops, double summations and double integrals, switching the order of the procedures may sometimes result in significant gain. For example, in [9], switching the order of a double summation significantly increased the computational efficiency of the intersection kernel support vector machine applied to computer vision. The essential idea of our fast algorithm is to switch the order of the nested for-loops, so that the repeated operations can be pushed out of some of them. The detailed pseudo-code is as follows, where the input data is stored in the variable *data* of N rows (genes) and M columns (samples).

```
1: for $i = 1 : M$ do
2:   % kernel distance between the $i^{th}$ data point and all other points $j$ for each gene $k$
3:   for $j = 1 : M$ do
4:     $SumMargin = zeros(N, 1)$
5:     for $k = 1 : N$ do
6:       $Dist(k, j) = KernelDistance(data(k, j), data(k, i));$
7:       $SumMargin(k) = SumMargin(k) + Dist(k, j);$
```

```
8:        end for
9:     end for
10:    % compute the $i^{th}$ entry of the first sum in Eqn (4) for all pairs of genes $k, l$
11:    for $k = 1 : N - 1$ do
12:       for $l = k + 1 : N$ do
13:          $SumJoint = 0;$
14:          for j = 1 : M do
15:             $SumJoint = SumJoint + Dist(k, j)Dist(l, j);$
16:          end for
17:          $MI(k, l) = MI(k, l) + log(\frac{SumJoint}{SumMargin(k)SumMargin(l)});$
18:       end for
19:    end for
20: end for
```

The above algorithm uses two strategies to improve the computational efficiency. One is to pre-compute the marginal probabilities for each gene, which avoids repeated calculations of the marginal probabilities inside the double for-loops in lines 11-12. This strategy is also used in ARACNE. The second strategy is to put the first summation in Equation (4) as the outer-most for-loop, instead of the double for-loop running over all pairs of genes. Using the second strategy, the kernel distance between any two particular data points is evaluated twice, which is significantly less than the N times repeats in ARACNE. This strategy is not currently used in ARACNE and is the key difference between ARACNE and our proposed algorithm.

It is clear that the computation time of the proposed algorithm is of complexity O(M^2N^2), which is the same as ARACNE [4, 10]. However, since the proposed algorithm performs much less repeated calculations, the computation time is significantly less than that of ARACNE. Simulation results are shown in Section 3.

The gain in computational efficiency comes at the price of a small amount of additional memory use, which is needed to stores the variables $Dist$ and

$SumMargin$. The additional memory use is of approximately the same size as the input data matrix.

In Matlab, some for-loops in the above pseudo-code can be written in more compact forms using matrix multiplications. The Matlab code of our algorithm is available at http://icbp.stanford.edu/software/FastPairMI/.

3 Results

We compare the computation time for obtaining the pairwise MI using ARACNE's Matlab and Java implementations downloaded from [4], and the our fast algorithm written in Matlab. The tests are performed using a personal computer (PC) with Intel Core 2 processor 2.66GHz. In Figure 1 (a), we fix the number of samples $M = 200$, vary the number of genes $N = 100, 200, ..., 1000$, and compare the computation time of ARACNE and the proposed fast algorithm in log-log scale. In Figure 1 (b), we fix the number of genes $N = 1000$ and vary the number of samples $M = 50, 100, ..., 500$. In Figures 1 (a) and (b), the curves for ARACNE and our fast algorithm share the same slope, meaning that the computation time of both methods grows at the same rate as the data size increases. Although the computational complexity is the same, due to significant reduction of repeated operations, the proposed fast algorithm achieves 15 to 89 times increase in speed compared with ARACNE, where greater gains in speed are achieved for data of larger size.

For a more realistic example, we examine the B-cell gene expression data analyzed in [4], which contains 9563 genes across 336 samples. The Java implementation of ARACNE takes 142 hours to compute the MI for all gene-pairs, whereas our fast algorithm takes 1.6 hours.

4 Conclusion

When analyzing large gene expression microarray data sets by information theoretic approaches, fast approaches are needed. We demonstrated a fast method for calculating the pairwise mutual information based on kernel estimation. Our method achieves significant gains in speed when compared to existing implementations, with modest increases in memory use. Moreover, our method is not limited to the analysis of gene expression data, and can be generally applied to speed up any algorithm that requires computing the mutual information matrix.

Acknowledgement

We gratefully acknowledge funding from the NCI Integrative Cancer Biology Program (ICBP), grant U56 CA112973.

Conflict of Interest

None declared.

References

[1] C. Chow, C. Liu, Approximating discrete probability distributions with dependence trees, IEEE Trans on Information Theory, 14(3) (1968), 462-467.

[2] A.J. Butte, L.S. Kohane, Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements, Pacific Symposium on Biocomputing, 4 (2000), 418-429.

[3] J.J. Faith, B. Hayete, J.T. Thaden, I. Mogno, J. Wierbowski, G. Cottarel, S. Kasif, J.J. Collins, T.S. Gardner, Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles, Plos Biology, 5(1) (2007), e8.

[4] A.A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, D.R. Favera, A. Califano, ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context, BMC Bioinformatics, 7(Suppl 1) (2006), s7.

[5] P.E. Meyer, K. Kontos, F. Lafitte, G. Bontempi, Information-theoretic inference of large transcriptional regulatory networks, EURASIP Journal on Bioinformatics and Systems Biology, 2007(1) (2007), 9 pages.

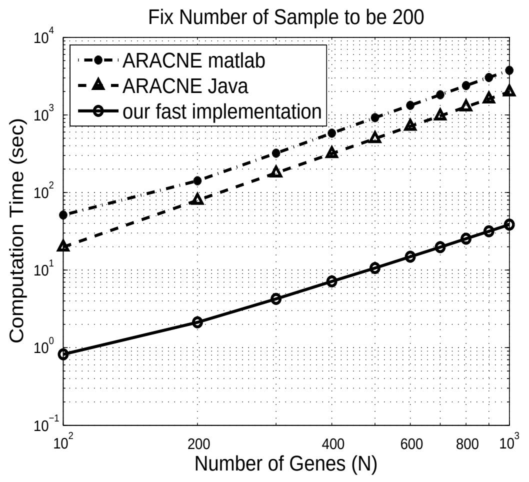
[6] C. Ding, H. Peng, Minimum redundancy feature selection from microarray gene expression data, Journal of Bioinformatics and Computational Biology, 3(2) (2005), 185-205.

[7] W. Zhao, E. Serpedin, E.R. Dougherty, Inferring connectivity of genetic regulatory networks using information-theoretic criteria, IEEE/ACM Trans on Computational Biology and Bioinformatics, 5(2) (2008), 262-274.

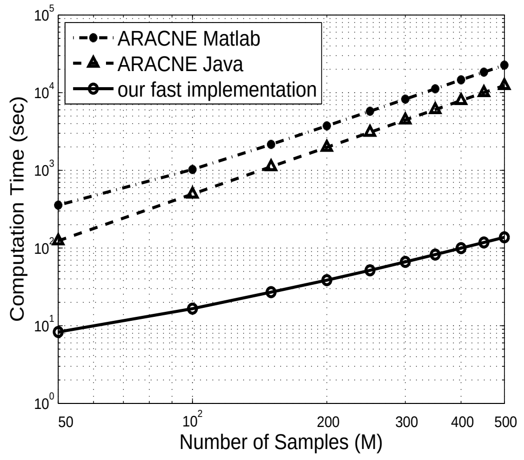
[8] J. Beirlant, E. Dudewicz, L. Gyorfi, E. van der Meulen, Nonparametric entropy estimation: An overview, International Journal of the Mathematical Statistics Sciences, 6(1) (1997), 17-39.

[9] S. Maji, A.C. Berg, J. Malik, Classification using intersection kernel support vector machines is efficient, IEEE Computer Vision and Pattern Recognition (CVPR), (2008).

[10] A.A. Margolin, K. Wang, W.K. Lim, M. Kustagi, I. Nemenman, A. Califano, Reverse engineering cellular networks, Nature Protocols, 1(2) (2006), 662-671.



(a)



(b)

Figure 1: Comparison of computation time between ARACNE and our fast algorithm. In Fig (a), the number of samples is fixed at 200 and the number of genes varies from 100 to 1000. In Fig (b), the number of samples varies from 50 to 500, and the number of genes is fixed at 1000.