Description of the PeakMatch Algorithm

Kazuya Haraguchi*

Version 2.0

Update History

January 31, 2019: The version 1.0 is published.

May 16, 2019: The version 2.0 is published: A simplest running example is added. The writing is polished.

---

1 Introduction

In this article, we present how to use the Python program of the PeakMatch algorithm that estimates the actual time when gene expressions in pseudo time based scRNA-seq data really happen. We also explain the algorithm and the adjustable parameters.

We applied the PeakMatch algorithm in our paper [3] and observed its effectiveness. The original source code is available at gitHub[1].

The article is organized as follows. In Section 2, we show how to run the program, providing a simplest running example. Those who would like to run the program immediately should refer to this section. In Section 3, we describe the problem of estimating pseudo times. In Section 4, we formulate the estimation problem as the the maximum weighted non-crossing matching (MWNCM) problem. In Section 5, we explain the detail of the algorithm, along with how to adjust the input parameters and the global variables in the source code.

2 How to Run the Program

To run the program, launch the terminal, and run `peakmatch.py` with some additional arguments as follows.

*As of writing, the author is with Faculty of Commerce, Otaru University of Commerce, Japan. E-mail: haraguchi@res.otaru-uc.ac.jp

[1] https://github.com/endo-lab/PeakMatch/blob/master/PeakMatch

```
$ python3 peakmatch.py sample_single.txt sample_bulk.txt 1 1 1 7
```

- By the 1st argument, you should specify the name of the file that contains scRNA-seq data. In the above example, it is `sample_single.txt`.
- Similarly, by the 2nd argument, you should specify the name of the file that contains cpRNA-seq data. In the above example, it is `sample_bulk.txt`.
- The subsequent arguments are a bit technical. See Section 5 for detail. If you are not interested in the detail, you may consider `1 1 1 7` as default values for the time being.

If you type the command appropriately and the data files are valid, the program will output the followings:

```
0 PT1 0
1 PT2 1 1
2 PT3 2 1
3 PT4 3 1
4 PT5 4 1
5 PT6 7.2 3.2
...
```

- The 1st column indicates the index of the pseudo time. The index begins from zero.
- The 2nd column indicates the label of the pseudo time.
- The 3rd column indicates the estimated actual time.
- The 4th column indicates the delay; in the above example, because PT5 is estimated as 4 and PT6 is estimated as 7.2, the delay shown in the PT6's row is $7.2 - 4 = 3.2$.

Usage. If you would like to see the usage, type simply:

```
$ python3 peakmatch.py
```

Format of a data file. Both data files should be in the tab-separated style. In each file, the gene labels should be written at the top row, and at the leftmost column, the time labels should be written. The expression levels should be given in the corresponding entries.

	(Gene 1's label)	(Gene 2's label)	...
(Time 0's label)	(exp. level)	(exp. level)	...
(Time 1's label)	(exp. level)	(exp. level)	...
⋮	⋮	⋮	...

3 Problem Description

Let Z be the set of whole genes under consideration. We discretize pseudo and actual times into integers for simplicity. We denote by $P = \{1, \ldots, m\}$ and $A = \{1, \ldots, n\}$ the sets of available pseudo and actual times, respectively. Suppose that, for each gene $z \in Z$, we are given pseudo time-series based scRNA-seq data $S_z = (s_{z,1}, \ldots, s_{z,m})$ and actual time-series based cpRNA-seq data $C_z = (c_{z,1}, \ldots, c_{z,n})$, where $s_{z,p} \in S_z$ and $c_{z,a} \in C_z$ represent the gene z's expression levels at a pseudo time p in the scRNA-seq data, and at an actual time a in the cpRNA-seq data, respectively.

To estimate the actual times of gene expressions in the scRNA-seq data, we would like to find pairs $(p, a) \in P \times A$ of pseudo and actual times so that the expression levels $s_{z,p}$ and $c_{z,a}$ are likely to be "comparable" for many genes $z \in Z$. Once such pairs (p, a) are found, we may estimate the actual time of $s_{z,p}$ by that of $c_{z,a}$.

The point is that, among the observed gene expression levels, "peaks" are the most important phenomena. Then it is desired that a peak in S_z and a peak in C_z should be matched. It is also demanded that the pseudo time order should be preserved in the time pairs. To be more precise, whenever a pseudo time p is matched to an actual time a, any pseudo time $p' > p$ should be matched to an actual time $a' > a$.

4 Formulation

We formulate the problem of finding such time pairs as the *maximum weighted non-crossing matching* (MWNCM) problem for a *bipartite graph*. The problem is polynomially solvable [1], meaning that it is efficiently solvable from viewpoint of the theory of computational complexity.

Preliminaries. A *graph* $G = (V, E)$ consists of a *vertex* set V and an *edge* set E, where each edge in E is a pair of vertices in V. An edge $e = (u, v)$ *joins* the vertices u and v, and the *extreme points* of e are u and v. The graph G is called *bipartite* if there is a partition $V = X \cup Y$ such that $E \subseteq X \times Y$ (i.e., each edge joins a vertex in X and a vertex in Y). A *matching* $M \subseteq E$ is a subset of edges such that no two edges share an endpoint in common. For an edge-weight function $w : E \to \mathbb{R}_+$, where $\mathbb{R}_+$ denotes the set of positive real numbers, we define the weight of M to be the sum of edge weights over M. For convenience, we represent the weight of M by $w(M)$ (i.e., $w(M) = \sum_{e \in M} w(e)$).

We assume that a bipartite graph is drawn in the manner of the 2-*layered drawing* [2]. Let $G = (X \cup Y, E)$ be a bipartite graph such that $X = \{x_1, \ldots, x_b\}$ and $Y = \{y_1, \ldots, y_d\}$. In the 2-layered drawing, taking two horizontal lines, we put $x_1, \ldots, x_b$ as distinct points on one line from left to right, put $y_1, \ldots, y_d$ as distinct points on the other line from left to right, and draw each edge as a straight line segment between extreme points. Let us take

two edges, say $e = (x_{i_1}, y_{j_2})$ and $e' = (x_{i_2}, y_{i_1})$. We say that e and e' *cross* or *intersect* if either ($i_1 < i_2$ and $j_1 < j_2$) or ($i_1 > i_2$ and $j_1 > j_2$) holds. A matching $M \subseteq E$ is called *non-crossing* if no two edges in M intersect.

Given a bipartite graph $G = (X \cup Y, E)$ and an edge-weight function $w : E \to \mathbb{R}_+$, the MWNCM problem asks for a maximum weighted non-crossing matching.

Composition of the graph. Suppose that we are given a function $f : P \times A \to \mathbb{R}_+ \cup \{0\}$ that evaluates the degree to which a pseudo time $p \in P$ and an actual time $a \in A$ are comparable, where f will be given concretely in the next section.

We construct a bipartite graph $G = (P \cup A, E_f)$ such that one vertex subset P of the bipartition is the set of pseudo times and the other vertex subset A is the set of actual times. In both vertex subsets, the vertices are ordered by the time order. The edge set E_f is defined along with the evaluation function f; we let $E_f = \{(p, a) \in P \times A : \ f(p, a) > 0\}$.

We have good reasons to estimate the actual times of pseudo times $p \in P$ based on an MWNCM M in $G = (P \cup A, E_f)$. Because M is a matching, if $(p, a) \in M$, the pseudo time p is matched to exactly one actual time a. Then it is natural to regard that the actual time of p is a. Moreover, because M attains the maximum weight, it must consist of peak pairs (p, a) that are evaluated as highly comparable. Due to the non-crossing constraint, the time orders are preserved. There may be a pseudo time p' that is not matched in M, but we can estimate its actual time in the interval between matched pseudo times.

5 Algorithm

Given a set Z of genes, a pseudo time-series based scRNA-seq data $\{S_z\}$ ($z \in Z$), an actual time-series based cpRNA-seq data $\{C_z\}$ ($z \in Z$), and the evaluation function $f : P \times A \to \mathbb{R}_+ \cup \{0\}$, the algorithm PeakMatch constructs the bipartite graph $G = (P \cup A, E_f)$, finds an MWNCM for it, and estimates the actual times of all pseudo times in P.

The algorithm consists of four parts: (i) preprocessing; (ii) construction of f and $G = (P \cup A, E_f)$; (iii) computation of an MWNCM; and (iv) time estimation.

(i) Preprocessing. For preprocessing, we organize scRNA-seq data $\{S_z\}$ and cpRNA-seq data $\{C_z\}$ ($z \in Z$) to make them more tractable.

Because scRNA-seq data are sometimes noisy, we take the exponential moving average of S_z for each $z \in Z$, to make the effect of noise smaller.

> **(Program note)** The radius and weight of the exponential moving average are set to 2 and 2^{-1}, respectively. These values are stored in the global variables `PSEUDO_RAD` and `PSEUDO_COEF` in the program code.

On the other hand, it is somewhat costly to collect cpRNA-seq data. For example, they could be observed only once in a couple of hours. Because n, the number of available actual times, must be too small, we extend $C_z = (c_{z,1}, \ldots, c_{z,n})$ by linear interpolation for each $z \in Z$.

(Program note) The number of inserted records can be adjusted by using the input parameter `inter`. For example, suppose that the actual times are taken for every four hours. By setting `inter` to 7, seven artificial records are inserted between actual times and thus the actual time between records is regarded as 30 minutes.

(ii) Construction of the evaluation function f and the bipartite graph $G = (P \cup A, E_f)$. For $(p, a) \in P \times A$, we evaluate how p and a are comparable, independently for each $z \in Z$. Specifically, we construct a function $f_z : P \times A \to \mathbb{R}_+ \cup \{0\}$ for each $z \in Z$ and define the total evaluation value $f(p, a)$ to be $f(p, a) \triangleq \sum_{z \in Z} f_z(p, a)$.

For each gene $z \in Z$, to determine $f_z(p, a)$, we decide whether or not the value $s_{z,p}$ (resp., $c_{z,a}$) is among a "peak area" in S_z (resp., C_z). We regard that $s_{z,p}$ (resp., $c_{z,a}$) is among a peak area if it is significantly larger than a general trend of $S_z = (s_{z,1}, \ldots, s_{z,m})$ (resp., $C_z = (c_{z,1}, \ldots, c_{z,n})$). The general trend of S_z (resp., C_z) is estimated by an exponential moving average, and $s_{z,p}$ (resp., $c_{z,a}$) is decided to be among a peak area if it is no less than the moving average plus $T\sigma$, where T is a real constant parameter and σ is the standard deviation of the difference between S_z (resp., C_z) and the moving average.

Let m' (resp., n') be the number of records in S_z (resp., C_z) that are identified among peak areas. For every $(p, a) \in P \times A$, if both $s_{z,p}$ and $c_{z,a}$ are among peak areas, we let $f_z(p, a) = 1/m'n'$, and otherwise, we let $f_z(p, a) = 0$.

From $f(p, a) = \sum_{z \in Z} f_z(p, a)$ and $E_f = \{(p, a) \in P \times A : \ f(p, a) > 0\}$, we can construct the bipartite graph $G = (P \cup A, E_f)$ immediately.

(Program note) The radius and weight of the exponential moving average are set to 50 and 1.1^{-1}, respectively. These values are stored in the global variables `INTV_RAD` and `INTV_COEF` in the program code.
T can be adjusted by using the input parameter `T`. A recommended value is 1.
By using the input parameter `intv`, we can choose whether we focus on all points in peak areas (1) or only on the maximum values (0).

(iii) Computation of an MWNCM. To find an MWNCM, we use the algorithm in [1] as a subroutine, which finds an MWNCM from $G = (P \cup A, E_f)$ in $O(|E_f| \log(|P| + |A|))$ time. See [1] for detail.

(Program note) We give a sufficiently large weight to the edge $(1, 1)$ (i.e., the edge between the first pseudo time and the first actual time) in order to match them forcibly.
For the edge (m, n) (i.e., the edge between the last pseudo time and the last actual time), using the input parameter `last`, we can choose whether we assign a sufficiently large weight to (m, n) (1) or not (0), to match them forcibly.

(iv) Time estimation. Let $M \subseteq E_f$ be the MWNCM that is obtained in (iii). For each $(p, a) \in M$, we estimate the actual time of p by a. For a pseudo time that is not matched to any actual time, we estimate its actual time as follows. Let p_{left} and p_{right} be pseudo times such that $p_{\text{left}} < p_{\text{right}}$ and that p_{left} and p_{right} are matched to actual times a_{left} and a_{right}, respectively (i.e., $(p_{\text{left}}, a_{\text{left}}), (p_{\text{right}}, a_{\text{right}}) \in M$). Let $p_1, \ldots, p_k$ be all pseudo times between p_{left} and p_{right} such that $p_{\text{left}} < p_1 < \cdots < p_k < p_{\text{right}}$. Then we estimate the actual time of a pseudo time p_t $(1 \leq t \leq k)$ by

$$a_{\text{left}} + \frac{t}{k+1}(a_{\text{right}} - a_{\text{left}}).$$

The estimated actual time could be fractional but we admit it here.

Let $p_{\max}$ be the rightmost pseudo time that is matched by M, that is, there is $a_{\max} \in A$ such that $(p_{\max}, a_{\max}) \in M$ but for all $p > p_{\max}$, no (p, a) belongs to M. For $p > p_{\max}$, we estimate its actual time by $a_{\max} + \epsilon(p - p_{\max})$, where ϵ is a sufficiently small constant.

(Program note) The constant ϵ is determined by a global variable `EPSILON` in the program code. The default value is 10^{-6}.

References

[1] Malucelli, F., Ottmann, T. and Pretolani, D.: Efficient Labelling Algorithms for the Maximum Noncrossing Matching Problem, *Discrete Applied Mathematics*, Vol. 47, 175–179 (1993).

[2] Sugiyama, K., Tagawa, S., Toda, M.: Methods for Visual Understanding of Hierarchical System Structures, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 11(2), 109–125 (1981).

[3] Torii, K., Inoue, K., Bekki, K., Haraguchi, K., Kubo, M., Kondo, Y., Suzuki, T., Takahashi, N., Shimizu, H., Kanesaka, Y., Uemoto, K., Fukuda, H., Araki, T. and Endo, M.: Circadian Clocks in Arabidopsis Regulates Cell Differentiation, in preparation.