BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–2

Supplementary material for "A generic motif discovery algorithm for sequential data"

Kyle Jensen[a], Mark Styczynski[a], Isidore Rigoutsos[a,b], Gregory Stephanopoulos[a*]

[a] Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA, [b] IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA

A simple, natural–language example

To demonstrate exactly how the algorithm works, we now provide a simple, natural–language example along with a description of the actions Gemoda would take at each step. Suppose we have a set of three words,

MOTIF

MOTOR

POTION

and we would like to find the motifs that some of these words share in common. Further, suppose that we are only interested in motifs that are at least four letters long and for which at least three of the four letters are "similar" between the windows. In this example, each word is a sequence, and the parameter L is 4. Thus, there are 7 possible windows that are taken sequentially from the three input sequences, numbered as shown in figure 1.

If we choose a similarity function based on the identity matrix with a threshold of three — that is, for two windows to be similar, at least three letters must be the same — then we find that only the following pairs of windows are similar: $(1,3)$, $(1,5)$, and $(2,6)$. Importantly, we note that though window 1 is similar to both windows 3 and 5, windows 3 and 5 are not similar to each other.

If, on the other hand, we choose a similarity function based on a matrix that distinguishes only between vowels and consonants — that is, any vowel is considered similar to any other vowel, and the same goes for any consonant — we would see different results for the same threshold value. In this case, we would find the following set of similarities: $(1,3)$, $(1,5)$, $(3,5)$, $(2,4)$, $(2,6)$, and $(4,6)$.

Given these similarity matrices for the different similarity functions, we can now cluster the graphs. Using the similarity matrix from the identity function, a clique–finding algorithm would find no cliques larger than size 2; that is, the only cliques that exist are the pairs of similar nodes. Since window 3 (MOTO) is not similar to window 5 (POTI), they cannot be in the same cluster.

However, if we use the similarity matrix produced by the weaker vowel/consonant function, we will find exactly two cliques of size 3: $\{1,3,5\}$ and $\{2,4,6\}$. Though there exist pairs of nodes that are similar, none of them is a clique because they are not maximal — that is, each individual pair of nodes that is similar (e.g., $(1,3)$) can have another node added to its set (5) without violating the pairwise similarity constraint, so only the larger set is a clique.

We also note that applying a transitive clustering function to the matrix created by the identity function would give still different results. In the transitive clustering function, the fact that windows 3 and 5 are not similar would not prevent them from being in the same motif; the function finds all disjoint subgraphs and defines them as the motifs. The motifs for such a case would be $\{1,3,5\}$ and $\{2,6\}$, which we will call motifs c_0^L and c_1^L, respectively.

Finally, we perform the convolution step. Using the last set of motifs described (with transitive clustering and the identity similarity function), we perform the convolution operation on each ordered pair of motifs; in this case, it means performing $c_0^L \curvearrowright c_1^L$, $c_1^L \curvearrowright c_0^L$, $c_1^L \curvearrowright c_1^L$, and $c_0^L \curvearrowright c_0^L$. For the first operation, we find the windows immediately after each of the windows in c_0^L, which is the set $\{2,4,6\}$. The intersection of this set with motif c_1^L is the convolved motif of length $L+1$, which is $\{2,6\}$; we can call this c_0^{L+1}. In performing $c_1^L \curvearrowright c_0^L$ and $c_1^L \curvearrowright c_1^L$, we note that no windows exist "after" windows 2 and 6, because their respective sequences end. In this case, the first set to be intersected is null, so the intersection is null. The final self–convolution operation also yields a null set. We now have only one motif for the new round of convolution, c_0^{L+1}. Performing $c_0^{L+1} \curvearrowright c_0^{L+1}$ results in a null set, meaning that there are no more motifs. At this point, we terminate convolution. It is worth noting that c_0^L is returned as a maximal motif because window 4 cannot be extended, but c_1^L is not because all of its instances were convolved in one direction.

Thus, we get different sets of motifs for different similarity and clustering functions. For identity similarity and clique–finding clustering, the final list of motifs is $\{\{\texttt{MOTIF},\texttt{POTIO}\},\{\texttt{MOTI},\texttt{MOTO}\}\}$. For identity similarity and transitive clustering, the final list of motifs is $\{\{\texttt{MOTIF},\texttt{POTIO}\},\{\texttt{MOTI},\texttt{MOTO},\texttt{POTI}\}\}$. For vowel/consonant similarity and either clustering method, the final list of motifs is $\{\{\texttt{MOTIF},\texttt{MOTOR},\texttt{POTIO}\}\}$.

Motif Significance

Each pair of nodes in a similarity graph can be described with two different quantities: $\eta_{i,j}$, the number of neighboring nodes (including each other) that the two nodes have in common, and $\chi_{i,j}$, the number of consecutive windows starting from each of those nodes that are connected to each other. For instance, if window 1 is similar to windows 1, 10, 25, and 36, and window 10 is similar to windows 1, 10, 25, and 37, then these two nodes have three neighbor nodes in common and $\eta_{1,10} = 3$. If window 1 is similar to 10, 2 is similar to

*to whom correspondence should be addressed

11, and 3 is not similar to 12, then there are two consecutive similar windows and $\chi_{1,10} = 2$.

By analyzing each node as above, we can accumulate a matrix of graph statistics, Φ, such that

$$\phi_{i,j} = |\{(x,y) : \eta_{x,y} = i, \chi_{x,y} = j, 0 \leq x, y \leq N\}| \quad (1)$$

(where the vertical bars indicate the cardinality of the set, or the number of ordered pairs) and

$$\Phi_{i,j} = \sum_{a=i}^{\infty} \sum_{b=j}^{\infty} \phi_{a,b} \quad (2)$$

These statistics can then be used in the following calculation for $p_{rel}(q,r)$, the relative likelihood of an output motif of length q and support r given the calculated similarity matrix:

$$p_{rel}(q,r) = \binom{N}{r} \left[\prod_{i=0}^{r-2} \left(\frac{\Phi_{i,1}}{\Phi_{i,0}} \right)^{r-i-1} \right] \left(\frac{\Phi_{r,q-L+1}}{\Phi_{r,1}} \right) \quad (3)$$

In this equation, the combinatorial factor represents the number of different ways that windows can be sampled in groups of r, the cumulative product represents the necessary conditions for the formation of a clique of length L, and the last factor represents the likelihood of extending a clique of support r to be length q. In this way, the relative likelihood measure attempts to represent the expected number of motifs of length q and support r that would occur at random given the calculated similarity matrix. Notably, this significance is based solely on the similarity matrix A, and so it can be used for either categorical or real–valued sequence data clustered with the clique–finding method.

Inductive proof of exhaustive maximality

When using clique–finding as the clustering function, each elementary pattern of length L is a clique in our similarity graph. That is, the elementary pattern is a set of windows that are all similar on a pairwise basis and there is no other window that can be added to the set.

When the algorithm enters the convolution stage, it starts by convolving each length L elementary motif with all of the others. An elementary motif that is *non–maximal* can be convolved with another elementary motif to yield a motif at level $L+1$ that has the same cardinality. All such motifs are marked as non–maximal. Those elementary motifs that remain unmarked cannot be extended on either side without losing support; since they are cliques we know they cannot be made greater in cardinality. Thus, all such unmarked cliques of length L can be labeled as maximal motifs and saved for output. In this way, we know that only maximal motifs will be returned to the user, and all such motifs will be returned.

When the "$\sqsubset$" operation is performed on two elementary motifs of length L that are being convolved, it ensures that no identical motifs of length $L+1$ exist and that no motif of length $L+1$ is a subset of any other. Additionally, since we have exhaustively compared a complete list of elementary motifs, and all such motifs are cliques with maximum cardinality, we are certain that all possible comparisons between motifs are being made. That is, no unique motifs of length $L+1$ could be created that are not subsets of motifs created by our exhaustive comparison. Finally, it is important to note that the result of convolving any two cliques will always be a clique. We know this because we take the set of all instances that can be extended (so the subgraph is maximal) and because all instances that are extended were pairwise similar in both windows being convolved (thus meeting our definition of similarity over multiple windows).

Thus, since Gemoda exhaustively generates *all possible* cliques of length $L+1$, and every added motif of length $L+1$ is maximal in support, we then know with certainty that c^{L+1} is an exhaustive list of motifs, or cliques, of length $L+1$. The induction step is then trivial, as setting L equal to $L+1$ at each step gives an exhaustive list of cliques just as when we started with c^L. This allows for a continual guarantee of exhaustiveness and maximality in output. The obvious termination condition for the algorithm is when $|c^i| = 0$. The following pseudocode sketch faithfully encapsulates the inductive algorithm described above.

```
begin
  n := 0
  while |c^n| ≠ 0 do
      for i := 0 to |c^n| step 1 do
          ismaximal := true
          for j := 0 to |c^n| step 1 do
              f := c_i^n ⌢ c_j^n
              if |f| ≠ 0
                  if f ⊏ c^{n+1} = false
                      c^{n+1} := c^{n+1} ∪ f
              else
                  choosemaximal(f, c^{n+1})
              fi
              if |f| = |c_i^n|
                  ismaximal := false
              fi
          fi
      od
      if ismaximal = true
          P := P ∪ c_i^n
      fi
  od
  n := n + 1
od
end
```

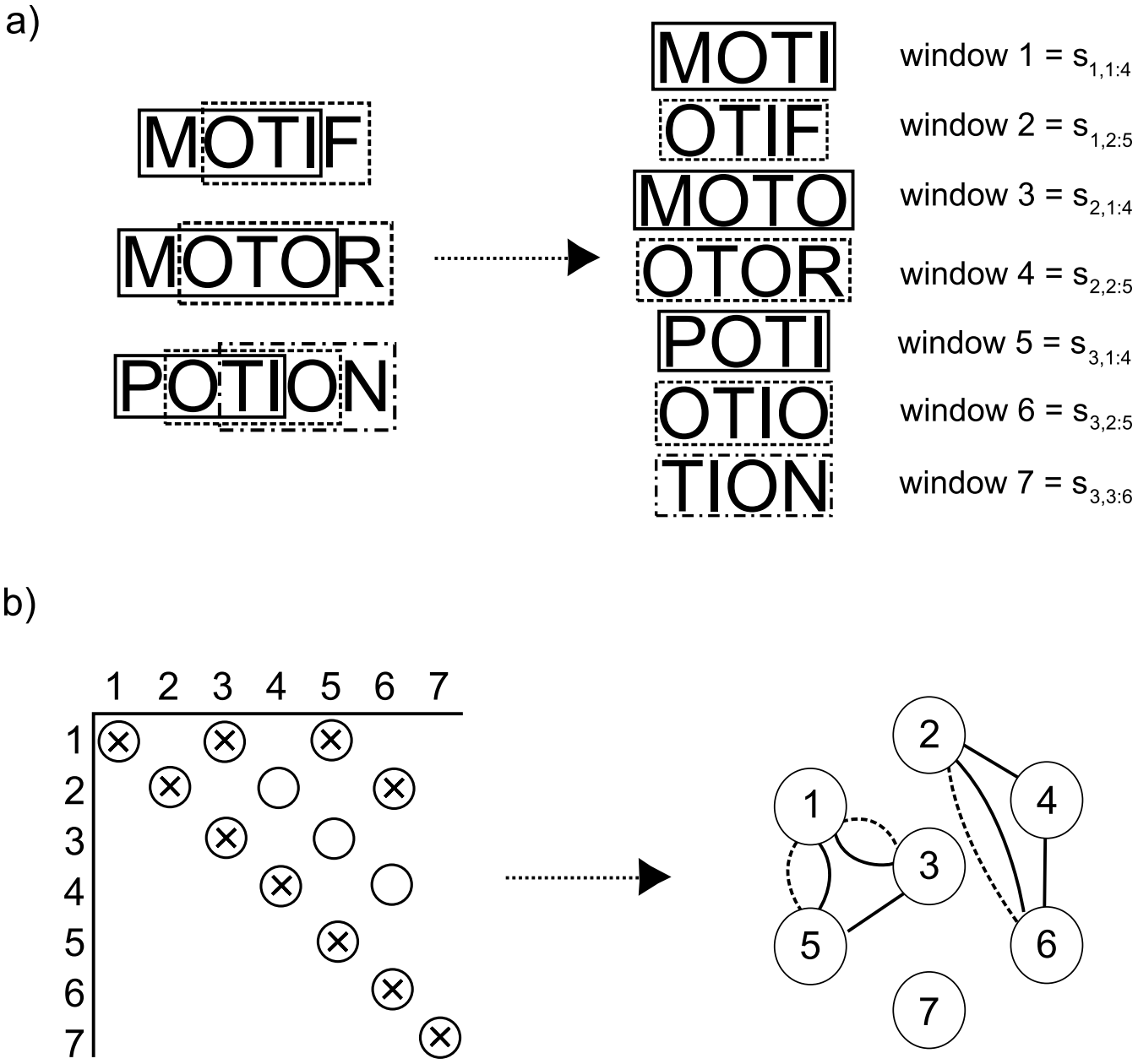

Fig. 1. A natural language example illustrating the steps that Gemoda takes. In a), we see the three words, or sequences, being broken into overlapping windows of four letters each. Gemoda would then compare each of these windows to each other using either of the similarity metrics described in the text. In b), we see the resulting similarity matrix and how it looks when drawn as a graph. In the matrix, two nodes are similar by the identity metric if there is an "X" at their intersection, while they are similar by the vowel/consonant metric if there is an "O" at their intersection. Making each window a vertex and connecting vertices with an edge if the windows are similar, we obtain the graph on the right. Dotted lines indicate similarity by the identity metric, while solid lines indicate similarity by the vowel/consonant metric. In this representation, it is clear what the results of both clique–finding and commutative clustering methods will be.