Sequence analysis

Sarment: Python modules for HMM analysis and partitioning of sequences

Laurent Guéguen

Laboratoire Biométrie et Biologie Évolutive, (UMR 5558); (NRS); Univ Lyon 1, 43 bd 11 Nov, 69622 Villeurbanne cedex, France

Received on February 18, 2005; revised on May 18, 2005; accepted on June 7, 2005
Advance Access publication June 9, 2005

ABSTRACT

Summary: Sarment is a package of Python modules for easy building and manipulation of sequence segmentations. It provides efficient implementation of usual algorithms for hidden Markov Model computation, as well as for maximal predictive partitioning. Owing to its very large variety of criteria for computing segmentations, Sarment can handle many kinds of models. Because of object-oriented programming, the results of the segmentation are very easy to manipulate.

Availability: Sarment is available on Linux, UNIX and MacOS X operating systems. It is under GPL license. There are binaries for python2.3 or python2.4, as well as sources, in the website http://pdb.univ-lyon1.fr/software/sarment

Contact: guéguen@biomserv.univ-lyon1.fr

SEQUENCE SEGMENTATION

Even though the drawbacks of using ‘sliding windows’ for sequence segmentation are well-known (such as the fixed size of the window), this method is currently the most used because its implementation is straightforward. Nonetheless, more precise segmentation methods have been developed over many years, and it is advantageous to render these methods easily usable.

A very popular way to segment a sequence is to model it with a hidden Markov Model (HMM), that is to say with a set of probabilistic states, usually corresponding to Markov models, all states being linked by probability transitions. With such modelling, it is possible to compute segmentations, thanks to the Viterbi and Forward-Backward algorithms (Rabiner, 1989). Since both algorithms have a linear time-complexity with the length of the sequence, HMMs are efficient tools for sequence analysis, and have been widely used for protein and genome segmentation, with many different Markovian models (Churchill, 1989; Peshkin and Gelfand, 1999; Bejerano et al., 2001; Nicolas et al., 2002; Boys and Henderson, 2004). In HMM, the constraint on the construction of the segments is the between-states probabilities; and an outcome of this modelling is that the lengths of the segments are required to follow sums of geometric distributions.

Maximal predictive partitioning (MPP) is a segmentation method that, given a set of models, builds the best partition of a sequence into $k$ segments (called a $k$-partition). MPP scores the adequacy of a model on a segment and during partitioning several models are compared, in order to optimally segment a sequence of letters into ‘homogeneous’ parts. Such a model can be Markovian, in which case, its prediction is the logarithm of its likelihood. Using dynamic programming, MPP computes the $k$-partitions for successive values of $k$ (Guéguen, 2001), using a linear time-complexity with $k$ and the length of the sequence. Hence, an MPP gives a multi-scale representation of the sequence, as shown in Figure 2. Choosing a specific number of segments corresponds to choosing a partition. Since an MPP can enumerate the best $k$-partitions for all possible values $k$ up to a maximum, a comparison of the segmentations obtained by MPP and HMM may be interesting; as for example, to observe the influence of the between-states probabilities on the HMM segmentations. Further, this comparison may help to understand how to interpret and use multi-scale partitionings of sequences.

Another way to compute multi-scale segmentations of sequences is to use a hierarchical process, the segments being recursively divided. Using simple composition criteria (such as C+G content), this approach has been applied to segmentation problems such as isochore detection or detection of CpG islands, and the corresponding programs are available (Li et al., 2002; Oliver et al., 2004). The criteria used by those programs to stop the segmentation process are based on the comparison of the composition of neighbouring segments: a testing level parameter is used to establish whether two segments have different compositions or not. The comparison of those methods and MPP would help to test the hierarchical constraint, or to obtain stopping criteria for MPP, and to extend hierarchical segmentation to more complicated criteria, such as the Markovian ones.

SARMENT

Sarment is a package of Python modules for easy building and manipulation of sequence segmentations. The first aim of Sarment is to provide an efficient implementation of the sliding windows method, HMM segmentation algorithms (Viterbi and Forward-Backward) as well as the MPP algorithm. As we want our program to work on very long sequences, such as whole chromosomes, data storage and heavy computing functions are written in C++.

The second aim of Sarment is to allow manipulation of the models that are used in HMMs and MPPs. There is a specific language for the prediction functions used for MPP, with numerous different criteria. Also, considering the fact that the set of models accepted by MPP includes Markovian models, the former are automatically translated into this language. It is, therefore, easy with this language to handle HMMs with numerous states and Markovian models with multiple orders. Moreover, there are classes to compute counts of sequences, pseudo-counts, proportions on words of heterogeneous lengths and to automatically build Markovian models.

Third, with Sarment, we want to provide a tool to easily handle sub-sequences, partitions and partitionings. This is the reason for
EXAMPLE OF SARMENT USAGE

As an example, we segment a sequence of the mouse genome to reveal the occurrences of CpG islands (Ponger and Mouchiroud, 2001). The sequence is 176 974 bases long. The several commands used for this example are available in the tutorial on the website.

As described previously by Durbin et al. (1998, p. 50), we use first-order Markov chains to model our data: two models are built by maximum likelihood on known data, the first for CpG islands and the other for the segments between CpG islands. Then, we build an HMM using the facts that CpG islands are on average 1000 bases long and that they are separated on average by 125,000 bases. With this model, the sequence is segmented using Viterbi and Forward-Backward algorithms (Fig. 1).

In Figure 2, we show the MPP up to a maximum of 50 segments of the sequence, with the models used in the previous HMM. The segments corresponding to the putative CpG islands appear successively in the $k$-partitions as the values of $k$ increase.

Externally to Sarment, we used several specific methods of CpG-island recognition, CpGproD (Ponger and Mouchiroud, 2001), CpGplot (Larsen et al., 1992) and CpG Island Searcher (Takai and Jones, 2002). Inside Sarment, we compared the resulting partitions of these methods with the ones computed previously. The resulting CpG islands detected are drawn in Figure 3. We can see that with these simply built Markovian models, the HMM and MPP partitions agree rather well with the CpG-tailored methods.

ACKNOWLEDGEMENT

The linkage between Python and the dynamic C++ libraries has been greatly eased by using the SWIG program (http://www.swig.org/).

Conflict of Interest: none declared.

REFERENCES


Guéguen,L. (2001) Segmentation by maximal predictive partitioning according to com-


All the modules of Sarment are described in the manual, which is available in pdf and html formats on the site. Some examples of manipulation of these classes are given in the tutorial, at the same location.

employed such a high-level programming language as Python. Hence, there are classes for data, models and all the elements in a segmentation. All objects are programmed in Python classes, which access C++ objects and methods via dynamic libraries. They are provided with many methods to make them easily manipulable, and can be easily improved upon.

All the modules of Sarment are described in the manual, which is available in pdf and html formats on the site. Some examples of manipulation of these classes are given in the tutorial, at the same location.

A row of arcs labelled by number $k$ represents the best $k$-partition. Each arc represents a segment. On each row, the relative height of an arc corresponds to the C+G density.

Fig. 3. CpG islands predicted using, from top to bottom, CpGproD, CpGplot, CpG Island Searcher, the 7-partition of the MPP, the Forward-Backward algorithm, and the 19-partition of the MPP.

employing such a high-level programming language as Python. Hence, there are classes for data, models and all the elements in a segmentation. All objects are programmed in Python classes, which access C++ objects and methods via dynamic libraries. They are provided with many methods to make them easily manipulable, and can be easily improved upon.

All the modules of Sarment are described in the manual, which is available in pdf and html formats on the site. Some examples of manipulation of these classes are given in the tutorial, at the same location.

EXAMPLE OF SARMENT USAGE

As an example, we segment a sequence of the mouse genome to reveal the occurrences of CpG islands (Ponger and Mouchiroud, 2001). The sequence is 176 974 bases long. The several commands used for this example are available in the tutorial on the website.

As described previously by Durbin et al. (1998, p. 50), we use first-order Markov chains to model our data: two models are built by maximum likelihood on known data, the first for CpG islands and the other for the segments between CpG islands. Then, we build an HMM using the facts that CpG islands are on average 1000 bases long and that they are separated on average by 125,000 bases. With this model, the sequence is segmented using Viterbi and Forward-Backward algorithms (Fig. 1).

In Figure 2, we show the MPP up to a maximum of 50 segments of the sequence, with the models used in the previous HMM. The segments corresponding to the putative CpG islands appear successively in the $k$-partitions as the values of $k$ increase.

Externally to Sarment, we used several specific methods of CpG-island recognition, CpGproD (Ponger and Mouchiroud, 2001), CpGplot (Larsen et al., 1992) and CpG Island Searcher (Takai and Jones, 2002). Inside Sarment, we compared the resulting partitions of these methods with the ones computed previously. The resulting CpG islands detected are drawn in Figure 3. We can see that with these simply built Markovian models, the HMM and MPP partitions agree rather well with the CpG-tailored methods.

ACKNOWLEDGEMENT

The linkage between Python and the dynamic C++ libraries has been greatly eased by using the SWIG program (http://www.swig.org/).

Conflict of Interest: none declared.

REFERENCES


Guéguen,L. (2001) Segmentation by maximal predictive partitioning according to com-


EXAMPLE OF SARMENT USAGE

As an example, we segment a sequence of the mouse genome to reveal the occurrences of CpG islands (Ponger and Mouchiroud, 2001). The sequence is 176 974 bases long. The several commands used for this example are available in the tutorial on the website.

As described previously by Durbin et al. (1998, p. 50), we use first-order Markov chains to model our data: two models are built by maximum likelihood on known data, the first for CpG islands and the other for the segments between CpG islands. Then, we build an HMM using the facts that CpG islands are on average 1000 bases long and that they are separated on average by 125,000 bases. With this model, the sequence is segmented using Viterbi and Forward-Backward algorithms (Fig. 1).

In Figure 2, we show the MPP up to a maximum of 50 segments of the sequence, with the models used in the previous HMM. The segments corresponding to the putative CpG islands appear successively in the $k$-partitions as the values of $k$ increase.

Externally to Sarment, we used several specific methods of CpG-island recognition, CpGproD (Ponger and Mouchiroud, 2001), CpGplot (Larsen et al., 1992) and CpG Island Searcher (Takai and Jones, 2002). Inside Sarment, we compared the resulting partitions of these methods with the ones computed previously. The resulting CpG islands detected are drawn in Figure 3. We can see that with these simply built Markovian models, the HMM and MPP partitions agree rather well with the CpG-tailored methods.

ACKNOWLEDGEMENT

The linkage between Python and the dynamic C++ libraries has been greatly eased by using the SWIG program (http://www.swig.org/).

Conflict of Interest: none declared.

REFERENCES


Guéguen,L. (2001) Segmentation by maximal predictive partitioning according to com-


