Modeling Processes Influenced by Hidden Processes

Greg Wimsatt
gwwimsatt@ucdavis.edu

Abstract:

Stationary processes have been the focus of our Natural Computation class. By studying processes dictated by different stationary processes at different times, we hope to expand our understanding of a sub-class of non-stationary processes, the so-called composite processes. We hope learn how to reconstruct the underlying composite machines behind some of these processes. We first introduce several simple types of composite processes and show how reconstructions of the composite machines can be easy if the correct type of composite process is assumed. We then focus on composite processes whose active stationary sub-process is always IID and switched very slowly. We present a method of determining which sub-processes were used and at roughly what times from a binary input sequence.

Introduction

In our Natural Computation class, we have primarily focused on stationary processes. Several methods of epsilon machine reconstruction can be used to handle many stationary processes. The difficulty of obtaining such an epsilon machine and its statistical complexity often correlate with the intuitive complexity of stationary processes. However, some non-stationary processes can be pretty easy to understand and yet may have epsilon machines which are very difficult to reconstruct with these methods.

To investigate some of these properties, we introduce the composite machines and their resultant composite processes, which are defined in the section titled **Background**. In brief, a composite machine consists of a primary machine that solely determines which of a number of secondary machines is active. The active secondary machine consequently outputs observed symbols during its time activated. In this paper, we focus on several simple types of composite machines to highlight some of these basic properties. Firstly, we discuss some composite machines whose active secondary machine switches after a fixed number of secondary output symbols. Composite machines whose active secondary switches after a variable, but sometimes small, number of secondary symbols are not explored in this paper. We show that it can be very difficult to reconstruct such processes with parse-tree reconstruction, but quite simple if we take the appropriate alternative reconstruction method. However, it is as yet unclear how to differentiate composite processes of different types that may require different reconstruction methods.

The remainder of the paper is spent considering composite machines whose primary machine outputs after a potentially variable, but always very large, number of secondary machine output symbols. We're motivated by the consideration of some simple cognitive functions to study these processes. For instance, if the reader were to look around outside she may notice trees, grass, and squirrels. Witnessing each of these individual objects might be represented by secondary processes. Seeing the object do something else or looking at another object may represent switching active secondary processes. Since seeing the individual objects requires perhaps trillions of photons to distinguish the object, each secondary process may need to be active far longer than the time required to receive one photon, which represents the time to receive each secondary output symbol, in order to comprehend the timing of each secondary processes and thus the resultant composite machine.

It is hoped that a slow enough primary process should allow enough time to get good samples of the secondary processes as they become active. We could then test chunks of data to see where new secondary states have probably been activated. A particular method is introduced to reconstruct such a composite machine given a large input string that is generated with secondary machines that are all IID over a binary alphabet. In this work, we focus on a description of an algorithm for the determination of which secondary machines were present and at roughly what times they were active.

More work needs to be done to take this secondary machine activation information and determine if any primary outputs were repeated. Segments where the primary outputs the same symbol twice in a row would result in a continuous activation of a single secondary machine, and so a simple list of secondary activations doesn't directly describe the full list of primary outputs. With a method of determining the complete list of primary outputs, the primary machine can be reconstructed and the full composite machine can be reconstructed as well.

Background

To get started, we better define a composite machine. Such a machine consists of a single machine, called the primary machine, and a set of machines, called secondary machines. All of these sub-machines individually generate stationary processes. At any given time, exactly one secondary machine is active. One directly observed symbol is generated by the active secondary machine for each unit of time for some length of time. The primary process then outputs a symbol that is not observed but corresponds to a particular secondary process. This secondary process becomes the new active process and the previous steps are repeated indefinitely. The net result is a potentially infinite string generated by a composite process. The time between primary machine outputs can be varied or fixed but must always be of positive integer values so that each active secondary process can complete a positive integer number of outputs. In this paper, we only consider machines where the secondaries have no memory of what state they were in after previous activations, and that output symbols from a binary alphabet.

We hope the reader finds composite machines to be rather intuitive: one primary machine determines which secondary machine to be active at any time and generate observed data. However, because of the potentially large times between switching active secondary machines, epsilon machine reconstruction can be far from trivial. Also, the resultant epsilon machines can have very large numbers of states and therefore have very large statistical complexities. Even if the primary machine outputs at every time step, resulting in a new active secondary machine with each time step, the epsilon machine can still be difficult to reconstruct with simple methods like parse-tree reconstruction, possibly resulting in an infinite mixed-state presentation where the causal states are not easy to ascertain.

Fixed-Rate-Primary Composite Machines

The first class of composite machines we will consider have primaries that output at every time step. This means that an active secondary is re-chosen with each observed output symbol.

Let it be noted that a fast IID primary with IID secondaries is equivalent to a simple IID process. To see this, realize that each secondary has a constant probability of being active and each output has a constant probability given any secondary that is active.

Fast Periodic Primary, 2 IID Secondaries

We consider composite machines with fast periodic primaries and two distinct IID secondaries over a binary alphabet. The active secondary therefore switches every time step to the other secondary. This composite machine is straight-forward to understand, but has a somewhat messy mixed-state presentation, as shown in Figure 1. For any finite future morph length used in the parse-tree reconstruction, the mixed-state presentation fails to completely capture the process. However, as we increase the morph length, the mixed-states farthest from the start state have output probabilities that approach those of the two actual secondary machines. So, looking closely at the mixed-state presentation, we can infer the two secondary IID processes to whatever accuracy we prefer by increasing the morph length.

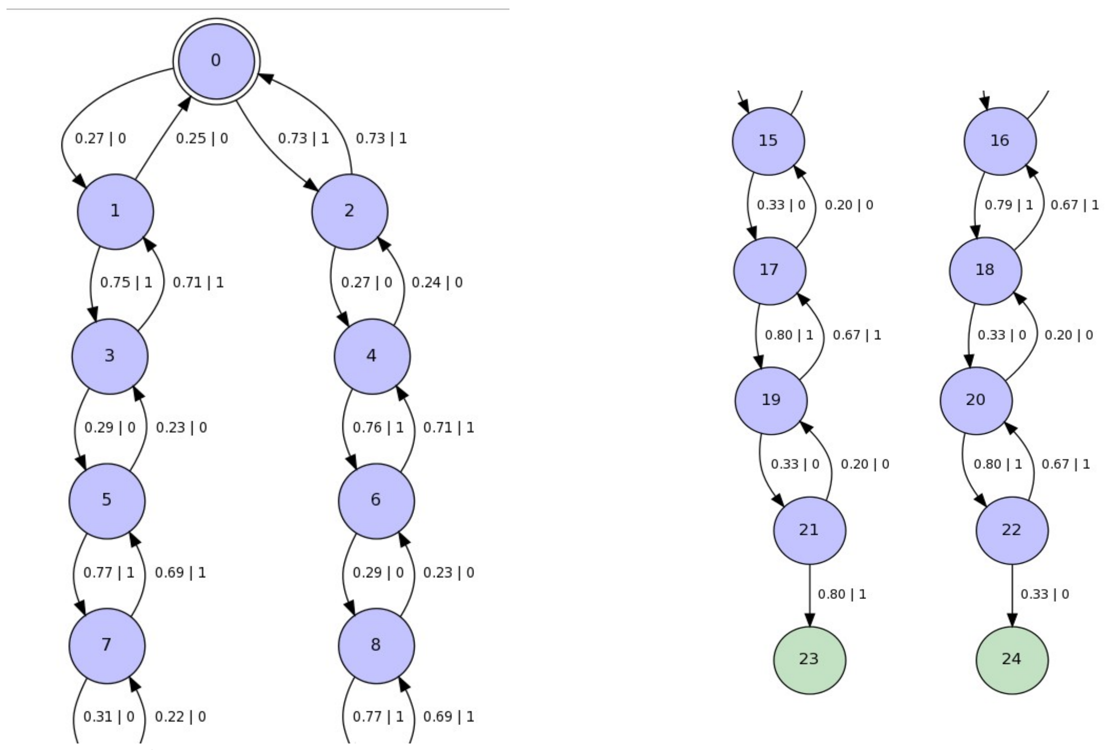

Figure 1: Mixed-state presentation of a periodic primary over two IID secondaries composite machine for L = 12. Here, the secondaries have probability 1/3 and 1/5, respectively, of generating a 0. The left panel shows the first few mixed-states. The right panel is shows the two mixed-state chains that began in the left panel extending to the final mixed-states. As L is increased, these end-chain mixed-states have output probabilities that approach those of the original secondary processes, and alternate appropriately.

If we instead reconstructed a composite machine by building a sub-machine for every even output symbol and one for every odd output symbol, we could readily identify the correct parameters of the actual secondaries that constructed the process. So if we suspected the process to be of the correct type of composite process, a periodic primary process over two IID secondaries, we could try to reconstruct it accordingly and readily interpret the results.

Fast Periodic Primary, 3 IID Secondaries

The point becomes more clear when we consider the fast periodic primary over three distinct IID secondaries. Again, this process is rather easy to understand. But, no matter what the parameters chosen for the probabilities of the secondary output symbols (as long as they are distinct), the mixed-state reconstruction, as in Figure 2, is quite a jumble. We can identify some mixed-states whose output probabilities can come arbitrarily close to the original secondaries by increasing the morph length, but it is much harder to identify them and it is unclear how to reliably ascertain which mixed-states are representative from the get-go. Furthermore, our mixed-state presentation now grows much faster with morph length, so parse-tree reconstruction seems unfeasible for obtaining arbitrarily accurate estimates of our original secondaries.

But, like for the fast periodic primary over two IID secondaries, we can reconstruct the correct composite machine if we know to try and make sub-machines over the right set of symbols. By reconstructing machines for symbols that are multiples of three away from each other in the input, we can readily capture the secondaries to as high precision as we have data.

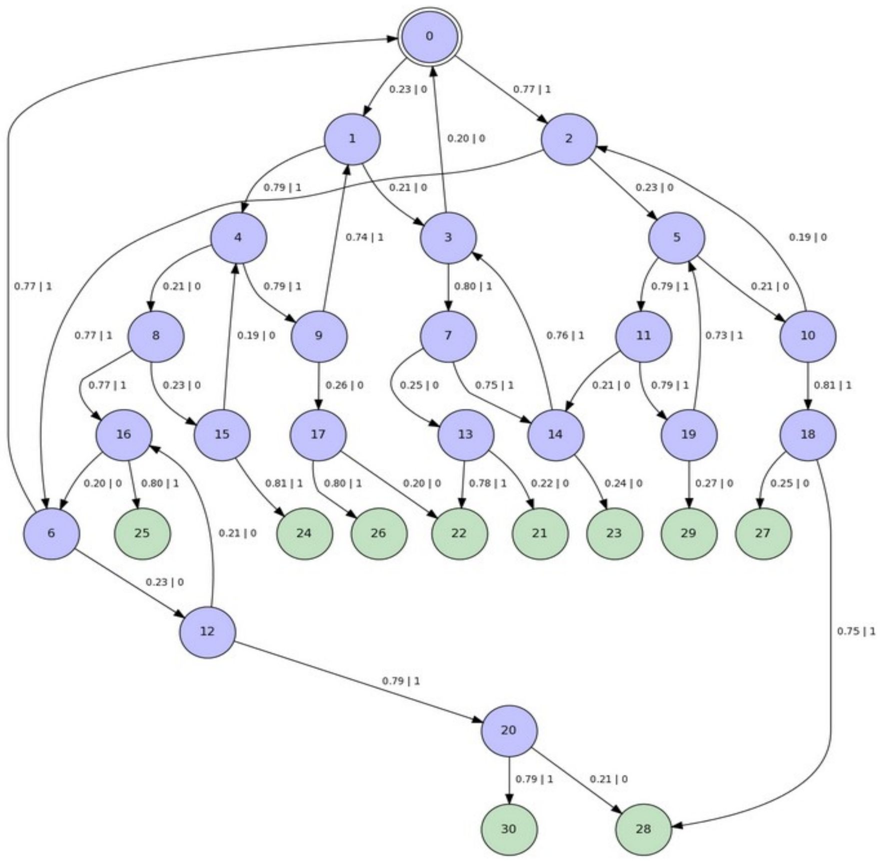

Figure 2: Mixed-state presentation of a fast periodic primary over three IID secondaries composite machine for L = 5. Here, the secondaries have probability 1/3, 1/5, and 1/7, respectively, of generating a 0. As L is increased, it looks like we can better take mixed-states whose output probabilities well resemble the original secondary processes, as in the fast periodic primary over two IID secondaries composite machine. However, this presentation is even messier and the becomes computationally straining with higher L values.

Other Fixed-Rate-Primary Composite Machines

The problem of parse-tree reconstruction for periodic primary composite machines grows as we consider more IID secondaries. However, as long as we know to try the correct periodicity for the primary machine (try the correct number of secondary machines), we can rebuild a fast periodic primary over an arbitrary number of IID secondaries.

For fixed-rate-primary composite machines in general, not just composite machines where the primary outputs every time step, the situation is similar. If we know the primary outputs a symbol every R secondary symbols that are output, we simply group observed adjacent symbols into blocks of R before grouping them again into periodically linked groups. However, we my still have troubles reconstructing the composite machine if we aren't aware of what phase of the first R block that the input symbols start in.

Putting aside issues with synchronizing the starting R-block of symbols, we could reconstruct many non-IID secondary processes by considering each of the R-blocks separately and parse-tree reconstruction with depth up to R. This widens the range of possible fixed-rate-primary composite machines we can analyze.

As a final point of possibility for fixed-rate-primary composite machines, those whose secondaries retain state information between activations can be similarly be parse-tree reconstructed. We would group symbols according to what we assume the primary output rate to be and the periodicity we assume it to be in, as before. But now we can expand the depth of the tree to any value less than the amount of symbols in a grouping, since all symbols in the group should be properly ordered and together constitute a stationary process.

It must be emphasized that the down side to these alternative analyses is that they require us to assume at least something about the form of the underlying composite process. If we don't know which type of composite process we are looking at, it may be hard to justify trying any of these methods. Crucially, it isn't clear how we can look at the results of these new methods and argue from them which base type of process the original composite process was probably in, let alone judge that it actually is a composite process.

Reconstructing Very Slow-Variable-Rate-Primary Over IID Secondaries

The main goal of the work is to reliably reconstruct composite machines with a variable rate, but always very slow, primary machine and IID secondary machines ("Very slow" refers to however slow is necessary for my algorithm to reliably reconstruct a given composite machine, which will be determined). This requires two main programs: one to generate the data for the base composite process, representing the composite machine, and one to reconstruct a composite machine from that data. At this point, regarding the latter, we have only developed an algorithm for reconstructing secondary machines and recording their activation times.

Program to Generate Data from a Composite Machine

The program for the base composite machine is a class that links a primary machine and a set of secondary machines. Both the primary and secondaries are instantiated as Mealy hidden Markov machines via the CMPy library, but were specifically built to model IID processes. The composite machine requires two bounds on the number of secondary output symbols seen before a primary symbol is output. Starting at the beginning of data generation and after every primary output, a specific number of secondary symbols to be output before the next primary output is generated by randomly taking an integer between the two bounds. This means that the processes generated were "synchronized" so that the primary will make its first output after a time consistent with it just having outputted.

Program for Secondary Machine Reconstruction and Activation Time Recording

The composite machine reconstructor as it stands has two crucial parameters that must be given to it. The first, L, is the minimum number of symbols to be considered in a chunk of data to be analyzed. Each chunk's total number of 1s divided by its total number of symbols gives that chunk its p value. If the chunk was generated by a single secondary process, the p value would be the best estimate of that secondary processes probability of generating a 1. Therefore, the higher the L value, the smaller the uncertainty on how well the p value represents the probability of obtaining a 1 from the chunk's parent secondary process. No uncertainties are calculated in this program, and the size of L is the sole control over this p value uncertainty.

It is assumed that the input data was generated from a composite machine whose primary machine outputs no faster than every 2*L secondary output symbols. Therefore, no chunk can have symbols from more than two different secondary processes.

The second important parameter, delta_p, is the maximum distance in p value that two chunks of data can have and still be considered “matching”. If two chunks were not matching, then it would be assumed that they are not representative of the same secondary process, or at least one of them has symbols from two secondary processes.

The first L symbols of an input-string are deemed representative of the first secondary process encountered and are chunked and called the past-chunk. The next L symbols are chunked and called the present-chunk. The two chunks are checked against each other to see if they match. If they do match, the past-chunk incorporates the first (oldest) symbol from the present-chunk and the present-chunk gains the next (2*L+1th) symbol of the input-string so that it continues to have L symbols. The two chunks are then checked again. If they match, the past-chunk again takes the first symbol of the present-chunk and the present-chunk gains the next (2*L+2th) symbol of the input-string so that it continues to have L symbols. This process is repeated until the input-string runs out or the past-chunk and present-chunk do not match. If the input-string runs out, the past-chunk's p value is the decided estimate of the first found secondary process's probability of emitting a 1. The program generates a name for this discovered secondary process and records its p-value, the number of 1s found in its chunk, and the number of symbols in its chunk, and records the input-string as having yielded one stationary process which began at time 0.

If the past-chunk and present-chunk were ever found to disagree, the past-chunk would then be declared the current best representative of the first found secondary process. The program would give it a name, record its p-value, the number of 1s found in its chunk, the number of symbols in the chunk, and that it occurred at time 0, as before. The end time of the present-chunk would also be noted and all symbols of the input-string up to this time would be discarded.

The process would restart as above but with the freshly trimmed input-string and using the noted time as the start time of the next found secondary process. This whole process continues again and again until the input-string runs out. However, for every secondary process found after the first, the program first checks to see if a previously recorded secondary process matches with the newly found one. Only if there are no matches will a new secondary process be declared and recorded. If a match is found, the two matching secondaries are merged. This means that the number of 1s and number of symbols found in the new chunk are added to those values recorded previously for the secondary process and a new p value is calculated for it. In this way, secondary processes are refined to become better representative as more data is taken in.

The net result is both a compilation of all reconstructed secondary machines, and a chronological list of their approximate activation times.

Presumably, transitions occur somewhere in the middle of present-chunks that are found to disagree with the past-chunk at the appropriate time. A transition occurring elsewhere would be indicative of either too big a delta_p, since an appropriate current-chunk wouldn't be different enough to stand out; or too small an L, since the uncertainty on how representative the p values are would be too low to accurately distinguish different secondary processes. Also, finding more switches between active secondary processes than actually occur should be fixed by increasing L. If L is deemed too large to be raised, delta_p can be increased to ensure the matching of fluctuations in p values that inevitably must occur in finite samples.

To finish the composite reconstructor, an algorithm must be devised that takes the list of reconstructed secondary machines and their activation times and solves for the underlying primary process. With the list of activation times, one could roughly determine the average secondary machine activation switch rate. However, it still isn't clear how to determine when the primary process outputs a particular symbol twice in a row, delaying a switch between active secondary machines. If the possible primary output rates are all very close to each other, the list of activation times would roughly look like it has one base unit of activation switch time, which would be observed when a primary output symbol is surrounded chronologically by different output symbols. For outputs of the same symbol multiple times in a row, rough multiples of this base unit of activation time would be observed. These different observations could be used to determine the true primary output rate, but a general solution isn't clear.

From a chronological list of all primary output symbols, any of the stationary process reconstruction methods could be used to determine the underlying primary process. In conjunction with a list of the corresponding secondary processes and the average transition time, the composite machine would be well estimated.

Conclusion

Composite machines offer a route into the study of non-stationary processes. The few simple cases described here demonstrate that such processes can be difficult to reconstruct with parse-tree reconstruction and yet be approachable on their own terms. However, a method of classifying composite processes into their appropriate general types is not clear.

The potential for modeling simple cognitive functions motivates the focus on composite processes with very slow primary processes. An algorithm for determining the secondary processes involved and their activation times has been devised. However, since a primary can output the same symbol twice (or more) in a row, it isn't trivial to go from the list of active secondaries and their active segments to a list of primary symbols and their complete output times. More work therefore needs to be done to reconstruct primary machines and thus full composite machines of even very slow primary output rates.