Learning Nonstationary Processes with Observable Operator Models

Lavinia Dinu

Computer Science
International University Bremen
Campus Ring 1
28759 Bremen
Germany

Type: Guided Research Report
Date: May 10, 2009
Supervisor: Prof. H. Jaeger

Executive Summary

Observable Operator Models (OOMs) are used in the field of Machine Learning for modelling stochastic processes, devised as a generalization of the widely known Hidden Markov Models (HMMs). They are an excellent alternative to HMMs to describe stationary symbol sequences due to the efficiency and accuracy of the existing learning algorithm, named Efficiency Sharpening (ES). However, an extension of this method to the class of nonstationary processes is yet to be fully completed. This task is of great importance, since it would introduce the power of OOMs to fields based on nonstationary processes, such as speech recognition or biosequence analysis. Research in this direction has been done by Dan Alistarh and Andrei Giurgiu in their thesis in Spring 2007 and 2008 respectively, namely finding a way to represent training data and extending the basic learning algorithm. My results introduce several possibilities of learning nonstationary sequences, including one based on the ES method, and investigate their efficiency.
Contents

1 Introduction  

2 Background  
   2.1 Preliminary Notions and Notation  
   2.2 Definition and Properties of OOMs  
   2.3 Characteristic Events and Interpretable OOMs  
   2.4 The Basic Learning Algorithm  
   2.5 Poor Man’s ES  

3 Statement and Motivation of Research  

4 Learning Nonstationary Processes  
   4.1 Representation of Finite Sequences  
   4.2 Training with Constant Length Inputs  
   4.2.1 Theoretical Considerations  
   4.2.2 Numerical Experiments  
   4.3 Extending OOMs  
   4.3.1 Definition  
   4.3.2 TOOMs as Predictors  
   4.3.3 Characteristic Events and the Basic Learning Algorithm  
   4.3.4 Poor’s Man ES  

5 Conclusions and Future Work  

3  

4  

7  

8  

9  

10  

11  

12  

13  

14  

16
1 Introduction

Observable Operator Models (OOMs) are mathematical models for stochastic processes, and have proven to be a strong alternative to other well-known tools such as Hidden Markov Models (HMMs). Their advantage over traditional methods lies in the elegance of the mathematical theory and the higher expressiveness: the theory surrounding OOMs can be expressed in terms of linear algebra and they can be used to model processes that can not be represented by HMMs.

The idea behind OOMs is to assign a probability value to each possible future observation sequence. These probabilities are then updated after an observation is made. Thus, the update transformation is identified with the observation, giving it the name observable operator (as seen in Figure 1). The insight is that for a large class of processes, namely linearly dependent processes, the models are finite and can be expressed by simple matrix algebra.

![Figure 1: The operator $\tau_a$ determines a transformation on the space of future distributions. From [1]](image)

In the case of stationary symbol sequences, several learning algorithms have been found. The basic versions are statistically inefficient, but the Efficiency Sharpening (ES) approach, an iterative method, is very efficient and yields accurate results. The purpose of this project is to work in the direction of finding a similarly successful algorithm for non-stationary symbol sequences.

The importance of this task is clear when thinking about the applications that would be made possible for OOMs, in fields that revolve around non-stationary processes. One example is speech recognition: an OOM can be 'taught' one-word inputs after being presented with multiple recordings of a single word pronounced by different people.
2 Background

This section will present the background notions regarding OOMs, using material from [1], [2], [3]. Only the case of stationary processes will be considered, and the corresponding results for the nonstationary case will be formulated in the following sections.

2.1 Preliminary Notions and Notation

Definition 2.1. Let $(\Omega, \mathcal{F}, P)$ be a probability space. A stochastic process is a collection

$$\{X_t \mid t \in T\}$$

of random variables $X_t$ defined on $(\Omega, \mathcal{F}, P)$, where $T$ is the index set of the process (usually referred to as 'time').

In the following, we will only consider discrete-time, finite-valued processes, so $T$ will be the integers. The sample space $\Omega$ will be that of the infinite strings over an alphabet $\Sigma$.

The elements of $\Sigma$ will be denoted with lowercase latin letters, possibly followed by an index, and the strings on $\Sigma$ will have an overline (e.g. $\overline{x}$). The empty string will be denoted $\varepsilon$. The set of all strings of length $k$ will be $\Sigma^k$, and the length of a string will be $|\sigma|$. The set of all strings over $\Sigma$ will be $\Sigma^*$. Concatenation will just be juxtaposition.

The probability $P(X_n = a_0, X_{n+1} = a_1, \ldots, X_{n+k} = a_k)$ will be written as $P_n(a_0a_1 \cdots a_k) = P_n(\overline{a})$. The conditional probability $P(X_n = a_0, \ldots, X_{n+k} = a_k) \mid X_{n-r+1} = b_0, \ldots, X_{n-1} = b_r)$ will be similarly denoted by $P_n(\overline{a} \mid \overline{b})$.

In the case of stationary processes, the index $n$ will be dropped, as the process is time-shift invariant. $n$ will also be dropped when it is equal to zero.

In order to measure the difference between two probability distributions (needed in the numerical experiments), the Kullback-Leibler divergence will be used. If $P$ and $Q$ are discrete probability distributions, then the KL divergence of $Q$ from $P$ is:

$$KL(P, Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}.$$

2.2 Definition and Properties of OOMs

OOMs can be defined by generalizing away from HMMs by relaxing several conditions on their structure.
A HMM specifies the distribution of a symbol sequence \((Y_n)_{n \in \mathbb{R}}\) where the random variables have outcomes in an alphabet \(O = \{a^1, \ldots, a^\alpha\}\). First, consider a Markov chain \((X_n)_{n \in \mathbb{R}}\) that produces sequences from a state set \(\{s_1, \ldots, s_m\}\). These states are the so-called hidden states. When the Markov chain is in the state \(s_i\), it produces an outcome \(a_j\) with probability \(P(Y = a_j | X = s_i)\). An example is given in Figure 2.

Figure 2: HMM with states \(s_1, s_2\) and initial state distribution \((2/3, 1/3)\). Small arrows indicate state transitions and emission probabilities are annotated along yellow arrows. From [2]

The HMM can be represented in a matrix formalism, by considering a stochastic matrix \(M\) that has at position \((i, j)\) the probability of transition from state \(s_i\) to \(s_j\), and for each \(a \in O\) a \(m \times m\) diagonal matrix \(O_a\) with the emission probabilities \(P(Y = a | X = s_j)\) on the diagonal. Denote by \(w_0\) the initial distribution of the Markov chain, and let \(T_a = M^T O_a, \mathbf{1} = (1, \ldots, 1)^T\). Then the well-known forward algorithm for obtaining probabilities of sequences for HMM can be written as [4]:

\[
P(a_0 \ldots a_r) = \mathbf{1} T_a \ldots T_{a_0} w_0.
\]

For OOMs, a different notation is used: \(M^T\) becomes \(\mu\) and \(T_a\) becomes \(\tau_a\). Now the definition of OOMs is obtained by allowing negative entries in all matrices and vectors.

**Definition 2.2.** A triple \(\mathcal{A} = (\mathbb{R}^m, (\tau_c)_{c \in \Sigma}, w_0)\), where \(w_0 \in \mathbb{R}^m\) and \(\tau_c : \mathbb{R}^m \to \mathbb{R}^m\) are linear operators is called an \(m\)-dimensional OOM over the alphabet \(\Sigma\) if the following conditions are satisfied:

1. \(w_0\) has a component sum of \(1\),
2. \(\mu := \sum_{c \in \Sigma} \tau_c\) has column sums of \(1\),
3. \(\forall a_0 \ldots a_r\) it is true that \(\mathbf{1} \tau_{a_r} \ldots \tau_{a_0} w_0 \geq 0\).

The last condition is obtained by adapting the forward algorithm from HMMs and ensuring that we don’t obtain negative probabilities. Then by computing the probabilities of finite length sequences by

\[
P(a_0 \ldots a_r) = \mathbf{1} \tau_{a_r} \ldots \tau_{a_0} w_0; \quad (1)
\]
$P$ can be extended to a probability measure on the set of right-infinite sequences over $O$ with the $\sigma$-algebra generated by finite sequences. This gives rise to a process which is stationary if $\mu w_0 = w_0$. A proof of this statement can be found in [3]. It is also possible to construct an OOM starting from a stochastic process. A formal treatment of this case is given in [1].

Using equation [1], OOMs can be used as generators to produce at time $n$ the symbol $a$ after $a_0, \ldots, a_{n-1}$ have been produced with probability $P(X_n = a|X_0 = a_0, \ldots, X_{n-1} = a_{n-1})$. For the first symbol, we need the vector $p_0 = (P(X_0 = a^1), \ldots, P(X_0 = a^s))^T$, where $s = |\Sigma|$. Using the notation:

$$\Sigma = \begin{pmatrix} 1\tau a^1 \\ \vdots \\ 1\tau a^s \end{pmatrix}$$

we obtain $p_0 = \Sigma w_0$.

Now note that:

$$p_n := P(a|\pi) = \frac{1\tau a \tau a_{n-1} \cdots \tau a_0 w_0}{1\tau a_{n-1} \cdots \tau a_0 w_0} = \frac{1\tau a}{1\tau a_{n-1} \cdots \tau a_0 w_0} \tau a_{n-1} \cdots \tau a_0.$$

Introducing the notation

$$w_{a_0 \ldots a_{n-1}} = w_\pi := \frac{\tau a_{n-1} \cdots \tau a_0 w_0}{1\tau a_{n-1} \cdots \tau a_0 w_0},$$

observe that

$$p_n = \Sigma w_\pi$$

and

$$w_{\pi a} = \frac{\tau a w_\pi}{1\tau a w_\pi}.$$

The vectors $w_\pi$ are called state vectors. Using these observations, we can use OOMs as generators in the following way:

**Algorithm 1** OOMs as Generators

1. $w \leftarrow w_0$
2. compute $\Sigma$
3. **loop**
4. $p \leftarrow \Sigma w$
5. generate symbol $a$ according to $p$
6. $w \leftarrow \frac{\tau a w}{1\tau a w}$
7. **end loop**

By making small modifications to the algorithm above, namely line 6, the OOM can be used to compute the probability that a certain sequence will be generated.
Because OOMs were obtained as a generalization of HMMs, any process described by a HMM can also be represented with OOMs. But the converse is not true. An example of a OOM process that can’t be described by HMMs is given in [2], the so-called probability clock. One of the operators \( \tau_a \) was considered to be a rotation with a non-rational angle, so that it is impossible to represent it using a matrix with non-negative entries. It is now obvious that OOMs are a proper generalization of HMMs, having more modelling power.

### 2.3 Characteristic Events and Interpretable OOMs

The basic learning algorithm for stationary processes uses the concept of interpretable OOMs. For these OOMs, there exist so-called characteristic events such that the vector of probabilities they will be realized in the future (after producing \( a_0 \ldots a_r \)) is exactly the state vector \( \tau_{a_r} \ldots \tau_{a_0} w \). By choosing random events, there is a high chance that these will actually be characteristic events, so it makes sense to exploit this property for the learning algorithm. In order to give a formal definition of these concepts, some further notation will be introduced.

Let \( P(B \mid \pi) := \sum_{\bar{b} \in B} P(\bar{b} \mid \pi) \), where \( B \) is a set of sequences. Given a state vector \( w \) and a sequence \( \bar{b} \) let \( P(\bar{b} \mid w) := 1_{\bar{b} w} \), the probability that the OOM will produce \( \bar{b} \) when started in state \( w \), and \( P(B \mid w) := \sum_{\bar{b} \in B} P(\bar{b} \mid w) \). Then:

**Definition 2.3.** Let \((X_n)_{n \geq 0}\) be an m-dimensional process with observables from \( O \) and for \( l \) sufficiently large, \( O^l = B_1 \cup \ldots \cup B_m \) be a partition of the set of strings of length \( l \). This partition is called a set of characteristic events if there are sequences \( \pi_1, \ldots, \pi_m \) such that the matrix \( (P(B_i \mid \pi_j))_{1 \leq i,j \leq m} \) is nonsingular.

Now it is possible to define interpretable OOMs:

**Definition 2.4.** Let \( B_1, \ldots, B_m \) be characteristic events for an m-dimensional process and let \( A \) be an OOM for that process. Then \( A \) is called interpretable with respect to \( B_1, \ldots, B_m \) if for all \( w \) which denote a state of \( A \), we have:

\[
w = (P(B_1 \mid w), \ldots, P(B_m \mid w))^T.
\]

It is possible to transform a given OOM into one that is interpretable with respect to certain characteristic events by applying a transformation to the original OOM. More details can be found in [1].

### 2.4 The Basic Learning Algorithm

In order to obtain a basic learning algorithm, two important properties of characteristic events are needed:

\[
w_0 = (P(B_1), \ldots, P(B_m))^T \quad (2)
\]
\[ \tau_a(P(\pi B_1), \ldots, P(\pi B_m))^T = (P(\pi a B_1), \ldots, P(\pi a B_m))^T \]  

Because probabilities of the form \( P(\pi B_i) \), where \( B_i \) is a characteristic event, can be estimated easily by simply counting the frequencies of the event from the training string, it is easy to obtain a good approximation of the operators \( \tau_a \), where \( a \in \Sigma \). It is sufficient to work only with indicative sequences \( s_i \), sequences such that vectors \( \{ P(s_i B_j) \}_{i=1}^{m} \) are linearly independent. We will denote the probability estimates of \( P(\pi B_i) \) obtained by counting with \( \hat{P}(\pi B_i) \). Using these insights, the basic algorithm is clear.

**Algorithm 2 Basic Learning Algorithm**

1: choose characteristic events \( B_1, \ldots, B_m \) and indicative sequences \( a_1, \ldots, a_m \) such that the matrix \( \hat{V} \leftarrow (\hat{P}(\pi_j B_i))_{1 \leq i,j \leq m} \) is nonsingular.

2: for all \( a \in \Sigma \) do
3: \( \hat{W}_a \leftarrow (\hat{P}(\pi_j a B_i))_{1 \leq i,j \leq m} \)
4: \( \hat{\tau}_a \leftarrow \hat{W}_a \hat{V}^{-1} \)
5: end for

The most important part of the algorithm is finding good characteristic events, such that the matrix \( \hat{V} \) is well-behaved with respect to inversion. A heuristic strategy has been found that gives quite good results. Let \( k \) be the length of the characteristic sequences, \( \kappa = |\Sigma|^k \). (\( \pi_j \)) \( 1 \leq j \leq \kappa \) and (\( \bar{b}_i \)) \( 1 \leq i \leq \kappa \) the alphabetical enumeration of \( \Sigma^k \).

Now construct a matrix \( V_0 \) of dimension \( \kappa \times \kappa \) that has at position \( (i, j) \) the number of times the string \( \pi_j \bar{b}_i \) appears in the training sequence. If the rows corresponding the same characteristic event are added together, the matrix \( V \) is obtained. So the idea of the procedure is to add together at each step two rows that have the highest pairwise correlation until a matrix of size \( m \times \kappa \) is obtained. The \( i \)-th characteristic event is then obtained by looking at which original rows have been joined in the \( i \)-th row.

Using this strategy and the basic learning algorithm, satisfactory models are obtained. They can be further used as initial models for the ES algorithm.

### 2.5 Poor Man’s ES

The ES method is an iterative method that uses an initial model \( \hat{A}^0 \) to iteratively obtain a sequence of improved models without relying on characteristic events. The model \( \hat{A}^n \) is used to improve the efficiency of the estimation procedure of the next model. A first variant of this, called a poor man’s ES in \( \Pi \), uses characterizers of length \( k \). A function \( C \) from \( \Sigma^k \) to the set of vectors with component sum 1 is called a characterizer if the state vector for some sequence \( s \) is \( C(P(v_0 | s) \ldots P(v_r | s)) \), where \( \{ v_i \}_{i=1}^{r} \) is the alphabetical enumeration of \( O^k \).

Another concept used is the reverse OOM \( A^r \), whose probability distribution \( P_{A^r}(s) = \)
$P_A(s')$, where $s'$ is the string $s$ reversed. It is obtained (proof in [1]) that the states of a reverse OOM obtained for sufficiently long reverse words are a characterizer of a forward OOM of the process, called the reverse characterizer. It has the property that it minimizes the variance of the models generated by a slightly modified version of the basic learning algorithm. The goal of the learning algorithm will then be to use an iterative procedure that finds a good estimation for this reverse characterizer.

The ES uses a heuristic algorithm to choose characteristic events for the initial model estimation and stores the corresponding characterizers. At each iteration, it updates the model using the reverse OOM and characterizer. On average, after 2-5 iterations the log-likelihood of the model stops to change, and the estimated models are more accurate than HMMs trained with the Baum-Welch method.

More advanced versions of this learning algorithm use suffix trees and adapts the indicative sequences to the training data. A description can be found in [1].

3 Statement and Motivation of Research

Although OOMs have been used with great success to model stationary stochastic processes (see [1]), this is not yet the case for non-stationary ones. This reduces the range of applicability of OOMs, leaving out important applications such as speech or handwriting recognition, biosequence analysis or diagnosis of heart diseases. That is why it is of prime importance to develop tools to deal with this particular kind of processes. So far, only an extension of the basic algorithm has been performed, but no numerical data was analyzed to investigate its performance. This project tries to rectify this, by investigating the performance of this existing algorithm and extending the poor man’s variant of ES.

An extra issue to be dealt with on the road to implementing efficient learning algorithms is modelling the training data. For non-stationary processes, these are usually many short sequences of variable length, while in the case of stationary processes, the data consist of a single long sequence. For example, when using OOM to recognize a handwritten word, many images representing that particular word written by different persons is presented to the OOM.

A problem that will not be attacked is the possibility of obtaining pseudo-OOMs, that give rise to negative probabilities when run. This issue is not yet solved for stationary processes and no possible solution is in sight. This problem will be inherited by OOMs for non-stationary processes, but it will not be tackled in this project.
4 Learning Nonstationary Processes

4.1 Representation of Finite Sequences

The solution employed by Dan Alistarh [5] for the problem of representing finite sequences was to use an extra symbol $\omega$ that marks the end of word. Thus, no major changes need to be made to the existing theory and implementation, but the OOM has to be hardcoded such that no symbol from the original alphabet can succeed $\omega$. Andrei Giurgiu [6] went in a different direction, by generalizing OOMs to accept finite sequences. These two approaches are however equivalent, because the $\omega$ symbol can just be omitted and the first model naturally becomes the second one.

However, in real-life data there are cases where the length of the sequences is approximately constant. Such an example would be handwritten symbols represented as images of constant dimension. For these instances, there is no need to hardcode an end symbol and tweak the OOM to stop at the fixed time. Therefore, some of the numerical experiments presented here will consider all inputs of finite length and will not compute an operator for the end symbol.

Nevertheless, there are cases where the length varies wildly; a randomly generated OOM will generate its first end symbol after varying (but short) time steps. In order to be able to account for this, the training sequences will be padded with $\omega$. In this case, my implementation will be mixture of both methods described in the first paragraph; the matrix $\mu$ will be extended to contain the probability values corresponding to $\omega$, but some changes will be made to the OOM structure: the generator algorithm will be adapted to stop when generating the end symbol. Storing values corresponding to $\omega$ is justified because the probability that the string ends would otherwise be retrieved by taking the difference to 1 of the column sum of $\mu$. It is not desirable however to perform this operation at each step.

4.2 Training with Constant Length Inputs

4.2.1 Theoretical Considerations

For nonstationary processes it is no longer possible to obtain estimates for $P(\pi)$ by counting its appearances in the input string. Because the probability distribution is time-dependent, only the appearances of $\pi$ at the beginning of the sequence are valid. The formula for the probability estimate becomes:

$$\hat{P}(\pi) = \frac{\# \text{appearances of } \pi \text{ at the beginning of a training sequence}}{\# \text{training sequences}}$$

Because the whole input needs to be taken into consideration, the length of the characteristic sequences has to be large. This is not feasible in terms of memory
requirements, as the counting matrix $V_0$ used to find the characteristic events grows exponentially. Furthermore, there is the possibility of obtaining zero operators for some symbol $a$ if it does not appear at the position $|\pi_i| + 1$, where $\pi_i$ is an indicative sequence. But if the number of training sequences is high enough, the probability of $a$ appearing at that position is close to 1.

Estimating $w_0$ is trivial using equation 2, because its elements are just $\hat{P}(B_i)$’s, where $B_i$’s are a characteristic events.

4.2.2 Numerical Experiments

In order to determine how well the training scheme performs, a series of experiments were run. A random nonstationary OOM was generated (such that $\mu w_0 \neq w_0$) and then it was used as a generator to produce several short strings. The concatenation of these strings was then fed to the ES algorithm and a first OOM $A$ was obtained. In parallel, the strings were used as training sequences for the algorithm described above and a second OOM $B$ was learnt. Then, for both OOMs, the KL divergence to the original model was computed, for both the training sequences and separately, some newly-generated test sequences. This experiment was repeated 10 times and the average values were taken in order to increase accuracy. All the code was built upon Prof. Jaeger’s toolbox[7].

Figure 3: KL divergence plots of OOMs trained with the same input sequences, an OOM $A$ using the ES method and $B$ using the nonstationary algorithm described in the text. The number of sequences was varied between 100 and 300 and $n$ denotes the length of the sequences.
The parameters that were varied in the various experiments were the number \( m \) and the length \( n \) of the sequences. Above are the KL divergence plots obtained.

One can see that the method used for the nonstationary OOM \( B \) performs better for each of the input parameters. The results for the test data are similar. However, the improvement over the normal learning method is not very dramatic and the algorithm is spatially inefficient due to the restriction of the length of characteristic sequences. A possible way to deal with these drawbacks is described in the next section.

### 4.3 Extending OOMS

#### 4.3.1 Definition

In order to overcome the problems described in the previous section, the structure of the OOMs will be changed. The \( \tau_a \) operators will be indexed with respect to time, so the definition and basic theory need to be changed accordingly. We begin by defining these time-indexed OOMs (TOOMs):

**Definition 4.1.** A triple \( A = (\mathbb{R}^n, (\tau^t_c)_{c \in \Sigma, t \in \mathbb{N}}, w_0) \), where \( w_0 \in \mathbb{R}^m \) and \( \tau_c : \mathbb{R}^m \rightarrow \mathbb{R}^m \) are linear operators is called an \( m \)-dimensional TOOM over the alphabet \( \Sigma \) if the following conditions are satisfied:

1. \( w_0 \) has a component sum of 1,
2. \( \mu^t := \sum_{c \in \Sigma} \tau^t_c \) has column sums of 1,
3. \( \forall a_0 \ldots a_r \) it is true that \( 1 \tau_{a_r} \ldots \tau_{a_0} w_0 \geq 0 \).

This definition can be used to define a probability measure.

**Proposition 1.** Let \( A = (\mathbb{R}^n, (\tau^t_c)_{c \in \Sigma, t \in \mathbb{N}}, w_0) \) be an TOOM. If one computes the probability of finite-length sequences over \( \Sigma \) by:

\[
P(a_0 \ldots a_r) = 1 \tau_{a_r} \ldots \tau_{a_0} w_0,
\]

\( P \) can be extended to a probability measure on the set \( \Omega \) of right-infinite sequences over \( \Sigma \) with the \( \sigma \)-algebra \( \mathcal{U} \) generated by finite sequences. This determines a stochastic process \((\Omega, \mathcal{U}, P, (X_n)_{n \in \mathbb{N}})\).

**Proof.** \( P \) can be uniquely extended to a probability measure if the following conditions are met:

- \( P(\bar{\omega}) \geq 0 \).

The third condition in the definition of OOMs guarantees that this always holds.
• $\sum_{\pi \in \Sigma^k} P(\pi) = 1$.

Because the column sum of $\mu^t$ is 1, it is true that $1 \mu^t v = 1 v$ for any vector $v$. The proof of this statement is trivial. Using this we have:

$$\sum_{\pi \in \Sigma^k} P(\pi) = 1 \sum_{t=0}^{k-1} \sum_{a \in \Sigma} \tau_a^t w_0 = 1 \sum_{t=0}^{k-1} \mu^t w_0 = 1 w_0 = 1.$$  

• $P(\pi) = \sum_{b \in \Omega} P(\pi b)$.

$$\sum_{b \in \Omega} P(\pi b) = \sum_{b \in \Omega} 1 \tau_b^{\pi} \tau_\pi w_0 = 1 \mu^{\pi} \tau_\pi w_0 = 1 \tau_\pi w_0 = P(\pi).$$

With this, the proof is complete.

From now on, the notation $\tau_\pi^l$ will be used for $\tau_a^0 \ldots \tau_a^l r$. The time index 0 will be usually dropped.

### 4.3.2 TOOMs as Predictors

Using the notation defined in the previous sections, the state vectors $w_\pi$ can be defined in the same way:

$$w_\pi = \frac{\tau_\pi w_0}{1 \tau_\pi w_0}$$

Define the matrix

$$\Sigma_t = \begin{pmatrix} 1 \tau_a^l \\ \vdots \\ 1 \tau_a^l \end{pmatrix}$$

The algorithm that generates sequences needs to be changed to account for finite sequences and the time index.

Algorithm 3 TOOMs as Generators

```
1: w ← w_0
2: n ← 0
3: repeat
4: compute $\Sigma_n$
5: p ← $\Sigma_n w$
6: generate symbol $a$ according to $p$
7: w ← $\tau_a^r w$
8: n ← n + 1
9: until a = $\omega$
```
4.3.3 Characteristic Events and the Basic Learning Algorithm

In order to learn a TOOM, each of the $\tau^t$ have to be learnt separately. Because $\tau^t_a$ is only applied at time $t$, one needs to only look at strings starting from that point. Consider all substrings starting at position $t$ of length $2e + 1$, where $e$ is the length of the characteristic events, and apply the basic learning algorithm described in section 4.2.1. This procedure enables spatially-efficient training with sequences of arbitrary length, as long the characteristic events remain short. But because of the window size, no probability estimates will be obtained for the final $2e + 1$ operators. Notice that the method of learning $w_0$ need not change; it can still be computed as the vector of probabilities of the characteristic events $(B^0_t)$.

Using the same experimental setup as before, we obtain a KL divergence several orders of magnitude times lower than the ones obtained in section 4.2.2, for both test and train data. This is a drastic improvement and, because the values are already very close to zero, already an efficient algorithm for estimating a TOOM.

This algorithm computes different characteristic events $(B^t_i)_{1 \leq i \leq m, t \in T}$ for each time step, so the question of defining an interpretable TOOM with respect to these characteristic events. The condition that it should satisfy is that the states $w$ of a TOOM $A$ have the property:

$$w = (P_t(B^t_1 | w), \ldots, P_t(B^t_m | w))^T.$$  

However, there is no clear way how to transform a given TOOM into one that is interpretable with respect to certain events.

A possible variant of the algorithm would be to compute the operators only at times $0, e, 2e, \ldots$ and assign the value of $\tau^t_a$ to all $\tau^t_{a+i}$, where $i = 1, \ldots, e - 1$. Initial experiments show that the results are comparable to the ones obtained without the time-skip, but the computations needed are fewer.

4.3.4 Poor’s Man ES

An idea for improving the performance of the algorithm would be to use an ES algorithm to compute the $\tau^t$’s instead of the basic one. One of course still needs to only take into consideration strings that start at the beginning of the current window, so the only change made to the algorithm would be the way the counting matrices are computed. The algorithm will then be:
Algorithm 4 Learning using ES

1: for all \( t \in T \) do
2:     compute matrix \( V^t_0 \) by looking only at strings that start at \( t \)
3:     find characteristic events \( B_1, \ldots, B_m \) using an heuristic method
4:     for all \( a \in \Sigma \) do
5:         compute matrices \( W^t_a \) by looking at words of the form \( a_j a B_i \) in the window \( (t - e, t + e) \)
6:     end for
7:     iterate until log-likelihood does not change too much
8: end for

An OOM of dimension 3 with an alphabet size of 3 was used to generate training data. The possible length of the sequences were 8, 10 and 12 and the number of sequences were 200, 250, 300, 400. The results of the experiment are plotted below.

Figure 4: KL divergence plots of OOMs trained with the same input sequences, multiplied by a factor of \( 10^5 \). \( n \) denotes the length of the sequence. Top: KL of training sequences; bottom: KL of test sequences. Details in the text.

This algorithm shows a significant increase in performance, especially for strings
of greater size. The OOMs obtained seemed well-behaved and didn’t generate negative probabilities.

5 Conclusions and Future Work

This paper gives several results in the context of extending learning algorithms for OOMs to nonstationary processes.

The first part tries to consider only one time-independent operator for each observable. Because of the nonstationarity condition, it is difficult to obtain correct estimates for the probability distribution of the process. A simple approach was outlined, that managed to give better results than learning with a classic OOM, but it is only applicable to sequences of very small length because of high memory usage.

In the second part, the observable operators are time-indexed, and at each time step a new operator has to be learned. This approach has given good results for training sequence generated by an existing OOM, but it is yet to be tested with real-world data. A possible issue would be the variable length of the input sequences; in some cases they would need to be scaled. For example, if inputs are the same word spoken at different speeds, it would be almost impossible for a time-indexed OOM to properly describe such a process without scaling.

A tweaked version of the ES was used in order to increase the efficiency of the training method; however, a proper ES still needs to be developed. One issue that would need to be approached is defining the reverse OOM and characterizer.

References


