Adaptive, Hands-Off Stream Mining

Spiros Papadimitriou Anthony Brockwell[1]
Christos Faloutsos[2]

December 2002

CMU-CS-02-205

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1]This material is based upon work supported by the National Science Foundation under Grant No. DMS-9819950.

[2]This material is based upon work supported by the National Science Foundation under Grants No. IIS-9817496, IIS-9988876, IIS-0083148, IIS-0113089, IIS-0209107 IIS-0205224 and by the Defense Advanced Research Projects Agency under Contract No. N66001-00-1-8936.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA, or other funding parties.

Keywords: streams, data mining, wavelets

Abstract

Sensor devices and embedded processors are becoming ubiquitous, especially in measurement and monitoring applications. *Automatic* discovery of patterns and trends in the large volumes of such data is of paramount importance. The combination of relatively limited resources (CPU, memory and/or communication bandwidth and power) poses some interesting challenges. We need both powerful and concise "languages" to represent the important features of the data, which can (a) adapt and handle *arbitrary* periodic components, including bursts, and (b) require little memory and a single pass over the data.

This allows sensors to *automatically* (a) discover interesting patterns and trends in the data, and (b) perform outlier detection to alert users. We need a way so that a sensor can discover something like "the hourly phone call volume so far follows a daily and a weekly periodicity, with bursts roughly every year," which a human might recognize as, e.g., the Mother's day surge. When possible and if desired, the user can then issue explicit queries to further investigate the reported patterns.

In this work we propose AWSOM (Arbitrary Window Stream mOdeling Method), which allows sensors operating in remote or hostile environments to discover patterns efficiently and effectively, with practically no user intervention. Our algorithms require limited resources and can thus be incorporated in individual sensors, possibly alongside a distributed query processing engine [CCC+02, BGS01, MSHR02]. Updates are performed in constant time, using sub-linear (in fact, logarithmic) space. Existing, state of the art forecasting methods (AR, SARIMA, GARCH, etc) fall short on one or more of these requirements. To the best of our knowledge, AWSOM is the first method that has all the above characteristics.

Experiments on real and synthetic datasets demonstrate that AWSOM discovers meaningful patterns over long time periods. Thus, the patterns can also be used to make long-range forecasts, which are notoriously difficult to perform automatically and efficiently. In fact, AWSOM outperforms manually set up auto-regressive models, both in terms of long-term pattern detection and modeling, as well as by at least $10\times$ in resource consumption.

1 Introduction

Several recent applications produce huge amounts of data in the form of a semi-infinite stream of values [GKMS01, GKS01, DGGR02, GG02]. Formally, a stream is a time sequence of numbers $X_0, X_1, \ldots, X_i, \ldots$ like samples or measurements at discrete time ticks.

Time sequences have attracted a lot of attention in the past [BJR94], for forecasting in financial, sales, environmental, ecological and biological time series, to mention a few. However, several new and exciting applications have recently become possible.

The emergence of cheap and small sensors has attracted significant attention. Sensors are small devices that gather measurements—for example, temperature readings, road traffic data, geological and astronomical observations, patient physiological data, etc. There are numerous, fascinating applications for such sensors and sensor networks, such as: (a) health care, with potentially wearable sensors monitoring blood pressure, heart rate, temperature etc. and detecting patterns and abnormalities, (b) industrial applications, keeping track of manufacturing process parameters and production, (c) civil infrastructure, with sensors embedded in bridges and highways [CGN00] monitoring vibrations and material deterioration, (d) road traffic conditions and safety, (e) smart houses and elderly care.

Although current small sensor prototypes [HSW$^+$00] have limited resources (512 bytes to 128Kb of storage), dime-sized devices with memory and processing power equivalent to a modern PDA are not far away. In fact, PDA-like devices with data gathering units are already being employed in some of the above applications (such as highway and industrial monitoring). The goal in the next decade is single-chip computers with powerful processors and 2–10Gb [CBF$^+$00, SGNG00] of nonvolatile storage.

Furthermore, embedded processors are becoming ubiquitous and their power has yet to be harnessed. A few examples of such applications are: (a) intelligent (*active*) disks [RFGN00] that learn common input traffic patterns and do appropriate prefetching and buffering, (b) intelligent routers that can monitor data traffic and simplify network management.

From now on, we use the term "sensor" broadly, to refer to any embedded computing device with fairly limited processing, memory and (optionally) communication resources and which generates a semi-infinite sequence of measurements.

The resource limitations unavoidably imply the need for certain trade-offs—it is impossible to store everything. Furthermore, we would like to make the most of available resources, allowing the sensor to adapt and operate without supervision for as long as possible.

This is the problem we address in this work. The goal is a "language" (i.e., model or representation) for efficient and effective, automatic stream mining. We want to collect information, in real-time and without any human intervention, to answer questions such as "what is the typical temperature pattern during a year" and "what is a good guess for the next Christmas' phone traffic?" Furthermore, we would ideally want to discover such patterns automatically and receive appropriate alerts.

This problem is orthogonal to that of continuous query processing. We focus on an adaptive algorithm that can look for arbitrary patterns and requires no *prior* human intervention to guide it. There are situations when we do not know *beforehand* what we are looking for. Furthermore, it may be impossible to guide the sensor as it collects data, due to the large volume of data and/or limited or unavailable communication. *If* further exploration is desired, users can issue targeted queries later on, guided by the general long-term patterns to quickly narrow down the "search space."

In detail, the main requirements are:

- **No human in the loop:** The method should not require human intervention before or during data gathering. In a general sensor setting we *cannot afford* human intervention.

- **Periodic component identification:** Humans can achieve this task, visually, from the time-plot. Our method should automatically spot multiple periodic components, each of unknown, *arbitrary* period.
- **Online, one-pass algorithm:** We can afford neither the memory or time for offline updates, much less multiple passes over the data stream.
- **Limited memory:** Sensor memory will soon be exhausted, unless our method carefully detects redundancies (or equivalently, patterns) and exploits them to save space. Furthermore, we ideally want our models to collect data even when network connectivity is intermittent (e.g., due to power constraints) or even non-existent.
- **Simple, but powerful patterns:** We need simple patterns (i.e., equations, rules) which can be easily communicated to other nearby sensors or to a central processing site. These patterns should be powerful enough to capture most of the regularities in real-world signals.
- **Any-time forecasting/outlier detection:** It is not enough to do compression (e.g., of long silence periods, or by ignoring small Fourier or wavelet coefficients). The model should be *generative*, in order to spot, store and report outliers: an outlier can be defined as any value that deviates too much from our forecast (e.g., by two standard deviations). It should be able to do so immediately, in real time.

Our AWSOM model has all of these characteristics, while none of the previously published methods (AR and derivatives, Fourier analysis, wavelet decomposition—see Section 2.1) can claim the same.

The rest of the paper is organized as follows: Section 2 discusses the related work. Section 3 briefly presents relevant background (some further information is contained in the appendices). Section 4 describes the proposed method and its algorithms. Section 5 presents experimental results on real and synthetic datasets. Section 6 gives the conclusions.

2 Related work

An interesting method for discovering *representative trends* in time series using so-called *sketches* was proposed by Indyk et. al. [IKM00]. A representative trend is a section of the time series itself that has the smallest sum of “distances” from *all* other sections of the same length. The proposed method employs random projections [JL84] for dimensionality reduction and FFT to quickly compute the sum of distances. However, it cannot be applied to semi-infinite streams, since each section has to be compared to all others.

Recent work by Gilbert et. al. [GKMS01] uses wavelets to compress the data into a fixed amount of memory, by keeping track of the largest Haar wavelet coefficients and updating them on-line (in the following, we will use the name *Incremental DWT* or *IncDWT* for short). The work in [GGI$^+$02a] presents this in the context of piecewise-constant histogram maintenance (see also [GKS02]). Also, [GGI$^+$02b] presents a novel method for approximate estimation of the largest Fourier coefficients by sampling only a portion of the series and proves bounds based on the uncertainty principle. However, all the above methods do not try to discover patterns and trends in the data. Thus, they cannot compete directly with our proposed method, which employs a *generative* model for pattern and trend detection.

More recently, Garofalakis et. al. [GG02] presented an approach for accurate data compression using *probabilistic wavelet synopses*. However, this method has an entirely different focus and cannot be applied to semi-infinite streams.

Further work on streams focuses on providing exact answers to pre-specified sets of queries using a minimum amount of memory. Arvind et. al. [ABB+02] study the memory requirements of continuous queries over *relational* data streams. Datar et. al. [DGI+02] keep *exact* summary statistics and provide theoretical bounds in the setting of a bitstream.

There is also some recent work on approximate answers to various types of continuous queries. Gehrke et. al. [GKS01] presents a comprehensive approach for answering correlated aggregate queries (e.g., "find points below the (current) average"), using histogram summaries to approximate aggregates. Dobra et. al. [DGGR02] present a method for approximate answers to aggregate multi-join queries over several streams, by using random projections and boosting.

Finally, a comprehensive system for linear regression on multi-dimensional time series data was presented very recently in [CDH+02]. Although this framework employs varying resolutions *in time*, it does so by straight aggregation, using pre-specified aggregation levels (although the authors discuss the use of a geometric progression of time frames) and can only deal with linear trends, using straight linear regression (as opposed to auto-regression).

2.1 Previous approaches

None of the continuous querying methods deal with pattern discovery and forecasting. The typical method for forecasting (i.e., generative time series modeling) uses the traditional *auto-regressive (AR)* models or their generalizations, *auto-regressive moving average (ARMA)*, *auto-regressive integrated moving average (ARIMA)* and *seasonal ARIMA (SARIMA)*—see Appendix A. Although popular, these methods fail to meet many of the requirements listed in the introduction. The most important failure is that they need human intervention and fine-tuning. As mentioned in statistics textbooks such as [BD91]:

> *"The first step in the analysis of any time series is to plot the data. [...] Inspection of a graph may also suggest the possibility of representing the data as a realization of [the 'classical decomposition' model]."*

Thus, such methods are not suited for remote, unsupervised operation. Furthermore, these methods have a number of other limitations:

- Existing methods for fitting models are typically batch-based, that is, they do not allow for recursive update of model parameters.

  While recursive least squares is effective in solving this problem for AR models, it does not generalize to handle the more general class of ARMA models.

- Established methods for determining model structure (i.e., period of seasonal components and order of the ARIMA model) are at best computationally intensive, besides not easy to automate.

- If there are non-sinusoidal periodic components, ARIMA models will miss them completely, unless *specifically* instructed to use a large window (for instance, Mother's day traffic will not be captured with a window less than 365 days long).

Large window sizes introduce further estimation problems, both in terms of resource requirements as well as accuracy. SARIMA models provide a workaround, but can deal only with one or, at best, a few *pre-determined*, *constant* periods.

- In addition, ARIMA models do not do a good job of handling "bursty" time series. ARIMA's linear difference equations eventually lead to to sinusoidal signals of constant or exponentially decreasing amplitude (or mixtures thereof). Thus, it can not generate signals with strong bursts, like, e.g., disk traffic [WMC+02] or LAN traffic [RMSCB99], even when these bursts are re-occurring (in fact, such bursts may even lead to exponential divergence in a generated sequence).

  While GARCH models introduced by [Bol86] provide a means of modeling the class of bursty *white noise* sequences, and a combined SARIMA-GARCH model can indeed model bursty time series with seasonal components, the computational difficulties involved with such models are far greater than those for the SARIMA models.

  Recently, the ARIMA model has been extended to *ARFIMA (auto-regressive fractionally integrated moving average)*, which handles the class of *self-similar* bursty sequences (such as *Fractional Gaussian Noise* [Ber94]). However, ARFIMA and its generalizations are even harder than ARIMA to use and require human expert intervention to determine yet another coefficient, the *Hurst exponent* [CB96].

All the above methods deal with linear forecasting. Non-linear modeling, using chaos and fractals, has been attracting increasing interest [WG94]. However, these methods also require the intervention of a human to choose the appropriate windows for (non-linear) regression or to configure an artificial neural network.

Data streams are essentially *signals*. There is a huge body of work in the signal processing literature related to compression and feature extraction. Typical tools include the Fast Fourier Transform (FFT) [OS75], as well as the Discrete Wavelet Transform (DWT) [PW00]. However, most of the algorithms (a) deal with *fixed length* signals of size N, and (b) cannot do forecasting (i.e., do not employ a *generative* model).

Thus the authors believe that there is a need (see also Table 2) for straightforward methods of time series model building which can be applied in real-time to semi-infinite streams of data, using limited memory.

3 Background material

In this section we give a very brief introduction to the most relevant background material (the appendices contain some further information).

3.1 Auto-regressive modeling

Auto-regressive models (also known as the *Box-Jenkins method* [BJR94]) are the most widely used. A more complete enumeration of AR models can be found in Appendix A. The main idea is to express X_t as a function of its previous values, plus (filtered) noise ϵ_t:

$$X_t = \phi_1 X_{t-1} + \ldots + \phi_W X_{t-W} + \epsilon_t \quad (1)$$

Symbol	Definition
X_t	Stream value at time tick $t = 0, 1, \ldots$.
N	Number of points *so far* from $\{X_t\}$.
$W_{l,t}$	Wavelet coefficient (level l and time t).
$V_{l,t}$	Scaling coefficient (level l and time t).
$\beta_{\delta l,\delta t}$	AWSOM coefficient, $(\delta l, \delta t) \in \mathcal{D}$).
$\mathcal{D}$	Window offsets for AWSOM (window "size" is $\lvert\mathcal{D}\rvert$).
$\mathrm{AWSOM}^{\lambda}(n_0, \ldots, n_\lambda)$	Number of level offsets (λ) and offsets per level $(n_0, \ldots, n_\lambda)$ in $\mathcal{D}$—see Definition 2. $(n_0, \ldots, n_\lambda)$ is also called the AWSOM *order*.
N_Λ, Λ, T	Parameters that determine the number of equations in the full model, $\mathrm{AWSOM}_{\Lambda,T}(n_0, \ldots, n_\lambda)$—see Definition 3.

Table 1: Symbols and definitions.

where W is a window that is determined by trial and error, or by using a criterion that penalizes model complexity (i.e., large values of W), like the *Akaike Information Criterion (AIC)*.

AR(I)MA requires manual preprocessing by trained statisticians to remove trends and seasonalities, typically by visual inspection of the sequence itself, as well as its *Auto-Correlation Function (ACF)*.

3.2 Recursive least squares

Recursive Least Squares (*RLS*) is a method that allows dynamic update of a least-squares fit. Updates can be done in $O(k^2)$ time and space, where k is the number of regression variables. More details are given in Appendix C and in [You84].

3.3 Wavelets

The N-point *discrete wavelet transform (DWT)* of a length N time sequence gives N wavelet coefficients. Each coefficient is responsible for a frequency range within a time window (the higher the frequency, the smaller the time window). Figure 1 shows the *scalogram*, that is, the "map" of the magnitude of each wavelet coefficient versus the location in time and frequency it is "responsible" for.

The DWT of a sequence can be computed in $O(N)$ time. Furthermore, as new points arrive, it can be updated in $O(1)$ amortized time. This is made possible by the structure of the decomposition into time and frequency (explained shortly) which is unique to wavelets. For instance, the Fourier transform also decomposes a signal into frequencies (i.e., sum of sines), but requires $O(N \lg N)$ time to compute and cannot be updated as new points arrive. A brief introduction to the wavelet transform can be found in Appendix B.

For our purposes here, we shall restrict ourselves to wavelets of the Daubechies family, which are easy to compute (even in the case of infinite streams), have desirable smoothness properties and successfully compress many real signals. In practice, although by far the most commonly used (largely due to their simplicity), Haar wavelets are too unsmooth and introduce significant artifacting [PW00]. In fact, unless otherwise specified, we use Daubechies-6.

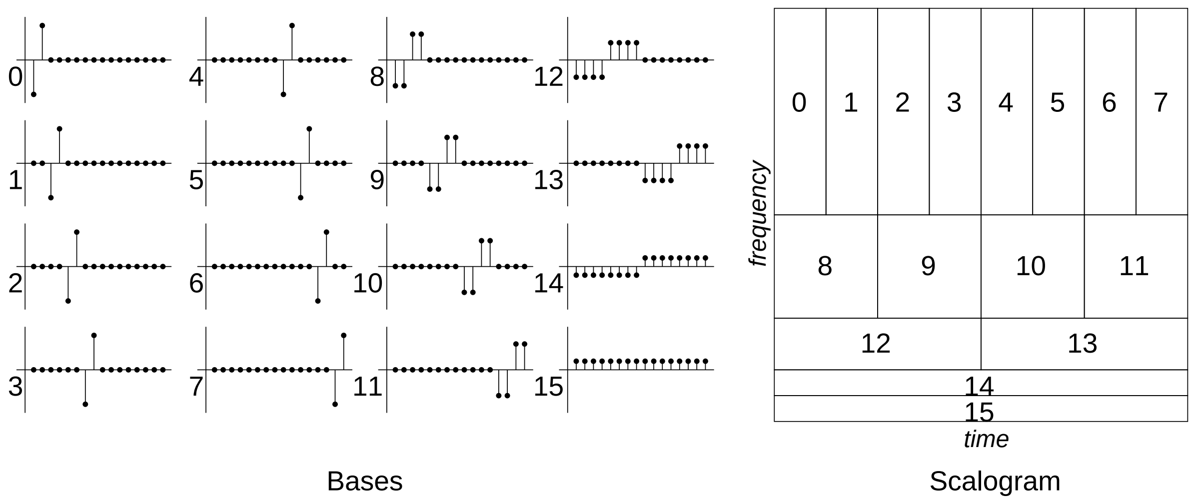

Figure 1: Haar bases and correspondence to time/frequency (for signal length $N = 16$). Each wavelet coefficient is a linear projection of the signal to the respective basis.

Incremental wavelets This part is a very brief overview of how to compute the DWT incrementally. This is the main idea of IncDWT [GKMS01], which uses Haar wavelets. In general, when using a wavelet filter of length L, the wavelet coefficient at a particular level is computed using the L corresponding scaling coefficients of the previous level. Recall that $L = 2$ for Haar, and $L = 6$ for Daubechies-6 that we typically use. Thus, we need to remember the last $L - 1$ scaling coefficients at each level. We call these the *wavelet crest*.

Definition 1 (Wavelet crest). *The* wavelet crest *at time t is defined as the set of scaling coefficients (wavelet smooths) that need to be kept in order to compute the new wavelet coefficients when X_t arrives.*

Lemma 1 (DWT update). *Updating the wavelet crest requires space $(L-1)\lg N + L = O(L \lg N) = O(\lg N)$, where L is the width of the wavelet filter (fixed) and N the number of values seen so far.*

Proof. See [GKMS01]. Generalizing to non-Haar wavelets and taking into account the wavelet filter width is straightforward. □

3.3.1 Wavelet properties

In this section we focus on some of the properties of the DWT which are relevant to AWSOM.

Time/frequency decomposition Notice (see scalogram in Figure 1) that higher level coefficients are highly localized in time, but involve uncertainty in frequency and vice-versa. This is a *fundamental* trade-off of any time/frequency representation and is a manifestation of the *uncertainty principle*, according to which localization in frequencies is inversely proportional to localization in time. This is a fundamental principle that implies certain trade-offs in *any* method! When dealing with semi-infinite streams in limited memory, we unavoidably need to make certain trade-offs. Given this, the wavelet representation is an excellent choice: it "compresses" well many real signals, while it is fast to compute and can be updated online.

Method	Contin. Streams	Trends / Forecast	Auto-matic	Memory
DFT (N-point)	NO	NO	—	—
SWFT (N-point)	YES(?)	NO	—	—
DWT (N-point)	NO	NO	—	—
IncDWT [GKMS01]	YES	NO	—	—
Sketches [IKM00]	NO	YES(?)	—	—
AR / ARIMA	YES	YES	NO [BJR94]	W^2
AWSOM	YES	YES	YES	$m\|\mathcal{D}\|^2$

Table 2: Comparison of methods.

Wavelets and decorrelation A wavelet transform using a wavelet filter of length $2L$ can decorrelate only certain signals, assuming their L-th order (or less) backward difference is a stationary random process [PW00]. For real signals, this value of L is not known in advance and may be impractically large: the space complexity of computing new wavelet coefficients is $O(L \lg N)$—see Lemma 1.

Wavelet variance One further benefit of using wavelets is that they decompose the variance across scales. Furthermore, the plot of log-power versus scale can be used to detect self-similar components (see Appendix B.1 for a brief overview).

4 Proposed method

In this section we introduce our proposed model.

4.1 Intuition behind our method

What equations should we be looking for to replace ARIMA's (see Equation 1)? In particular, how can we capture periodic components of arbitrary period?

First part—information representation As explained in section 3.3.1, given our limited resources, we have to make a choice about how to efficiently and effectively capture the important information in the sequence. Traditional models (such as ARIMA) operate directly in the time domain. Thus, they cannot deal with redundancies, seasonalities, long-range behavior, etc. This is where a human expert is needed to manually detect these phenomena and transform the series to match ARIMA's assumptions.

This is a crucial choice—is there a better one? We want a powerful and flexible representation that can adapt to the sequence, rather than expect someone to adapt the sequence to the representation. Wavelets are extremely successful in compressing most real signals, such as voice and images [Fal96, Fie93, PTVF92], seismic data [ZdZ98], biomedical signals [Aka97] and economic time sequences [GSW01].

By using wavelet coefficients, we immediately discard many redundancies (i.e., near-zero valued wavelet coefficients) and focus on the things that really matter. Furthermore, the DWT can be computed quickly and updated online.

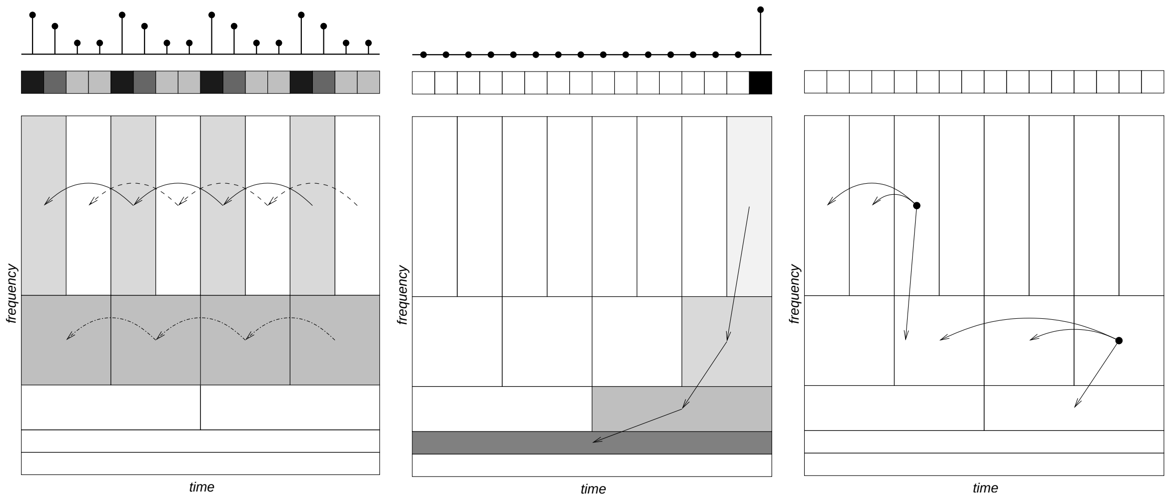

Figure 2: AWSOM—Intuition and demonstration. AWSOM captures intra- and inter-scale correlations. Also, the left figure demonstrates why we fit different models per level.

Second part—correlation structure In the wavelet domain, how can we capture arbitrary periodicities? The answers come from the properties of the DWT. A periodic signal (even a spike train—e.g., Mother's day traffic), will have high absolute values for the wavelet coefficients at the scales that correspond to its frequency. Even more, successive coefficients on the same level should have related values (see Figure 2, left). Thus, in order to capture periodic components, we should look for correlations between wavelet coefficients within the same level.

Furthermore, how should we capture bursts? Short bursts carry energy in most frequencies. Therefore wavelet coefficients across different scales (i.e., frequencies) will have large values (see Figure 2, middle). If the phenomenon follows some pattern, then it is likely that there will be an inter-scale correlation among several of the wavelet coefficients.

Third part—correlation modeling The last question we need to answer is: what type of regression models should we use to quantify these correlations? Our proposed method tries to capture inter- and intra-scale correlations by fitting a linear regression model in the wavelet domain. These can also be updated online with RLS.

To summarize, we have argued for using the wavelet representation of the series and capturing correlations within and across scales (see Figure 2, right). We have "substituted," in effect, the human expert with the DWT and this correlation structure. Thus, we expect that linear regression will successfully capture these correlations with *very few* coefficients.

UpdateCrest ($X[t]$)**:**	**Update** ($X[t]$)**:**

Foreach $l \geq 0$ s.t. 2^l divides t:
  Compute $V[l, t/2^l]$
  If 2^{l+1} divides t:
    Compute $W[l, t/2^{l+1}]$
    Delete $W[l, t/2^{l+1} - L]$
	UpdateCrest($X[t]$)
	Foreach new coefficient $W[l, t']$ in the crest:
	  Find the linear model it belongs to
	   based on l and $t' \mod \Lambda$
	  Update $X^T X$ and $X^T y$ for this model

Figure 3: High-level description of update algorithms.

4.2 AWSOM modeling

Formally, our proposed method tries to fit models of the following form:

$$\boxed{\begin{array}{l} W_{l,t} = \qquad\qquad \beta_{0,1} W_{l,t-1} + \beta_{0,2} W_{l,t-2} + \dots \\ \qquad \beta_{1,0} W_{l-1,t/2} + \beta_{1,1} W_{l-1,t/2-1} + \dots \\ \qquad \beta_{2,0} W_{l-2,t/4} + \dots \\ \qquad\qquad \dots \end{array}}$$

or more concisely (where $\epsilon_{l,t}$ is the usual error term)

$$W_{l,t} = \sum_{(\delta l, \delta t) \in \mathcal{D}} \beta_{\delta l, \delta t} W_{l+\delta l, t/2^{\delta l} - \delta t} + \epsilon_{l,t} \tag{2}$$

where $\mathcal{D}$ is a set of index offsets. The $\beta_{\delta l, \delta t}$ are called the AWSOM *coefficients*.

Definition 2 (AWSOM **order).** *The set of offsets is always of the form*

$$\begin{array}{rl} \mathcal{D} = \{ & \qquad (0,1),\ (0,2),\ \dots,\ (0,n_0), \\ & (1,0),\ (1,1),\ (1,2),\ \dots,\ (1, n_1 - 1), \\ & \ \dots, \\ & (\lambda, 0),\ \dots,\ (\lambda, n_\lambda - 1)\ \} \end{array}$$

i.e., each wavelet coefficient is expressed as a function of the previous n_0 wavelet coefficients on the same level, n_1 coefficients from one level below and so on. For a particular choice of $\mathcal{D}$, we use $\mathrm{AWSOM}^\lambda(n_0, \dots, n_\lambda)$ *or simply*

$$\mathrm{AWSOM}(n_0, n_1, \dots, n_\lambda)$$

to denote this instance of our model. We call $(n_0, \dots, n_\lambda)$ the order *of the* AWSOM *model. The* total order *is the number of* AWSOM *coefficients k per equation, i.e., $k = \sum_{\delta l = 0}^{\lambda} n_{\delta l}$.*

A fixed choice of $\mathcal{D}$ is sufficient for all signals. In most of our experiments we use $\mathrm{AWSOM}(6, 4, 2)$ ($k = 12$).

The naive approach would be to fit Equation 2 to *all* data points (i.e., wavelet coefficients). However, an approach that gives more accurate results is to fit one equation per level (see Figure 2), as long as the level contains enough wavelet coefficients to get a good fit. Thus, in actual use on a running stream, we would fit one equation for every level $l < \Lambda$, where Λ is the level that has no more than,

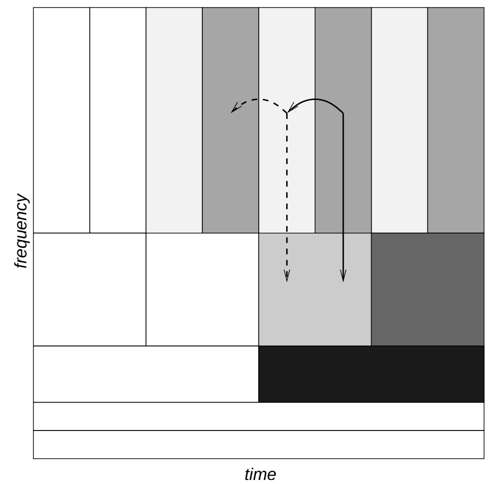

Figure 4: Illustration of $\mathrm{AWSOM}_{2,2}(1,1)$ with $N_\Lambda = 2$. The shade of each wavelet coefficient corresponds to the model equation used to "predict" it. The unshaded wavelet coefficients correspond to initial conditions (i.e., with incomplete AWSOM window $\mathcal{D}$).

say, $N_\Lambda = 16$ wavelet coefficients. For levels $l \geq \Lambda$ we can either keep the exact wavelet coefficients (which would be no more than $16 + 8 + \cdots + 1 = 31$ in the above case) and/or fit one more equation. Thus, as more data points arrive, the value of Λ gradually increases as necessary.

Besides fitting one equation per level (up to Λ), when we use AWSOM^λ with $\lambda \geq 1$, we also fit different equations depending on time location t. For instance, if we are using $\mathrm{AWSOM}^1(n_0, 2)$, we should fit one equation for pairs $W_{l,2t}$ and $W_{l-1,t}$ and another for pairs $W_{l,2t+1}$ and $W_{l-1,t}$ (see Figure 4). In general, we need 2^λ separate models to ensure that the inter-scale correlations λ levels down are not "shoehorned" into the same regression model.

To summarize, the full AWSOM model fits a number of equations:

$$W_{l,t} = \sum_{(\delta l, \delta t) \in \mathcal{D}} \beta^{l',t'}_{\delta l, \delta t} W_{l+\delta l, t-\delta t} \epsilon_{l,t} \tag{3}$$

where $l' \equiv \min(l, \Lambda)$ and $t' \equiv t \bmod T$. For example, if $T = 2$ and $\Lambda = 1$, we estimate one linear equation for each set of wavelet coefficients $W_{0,2i}$, $W_{0,2i+1}$, $W_{l,2i}$ and $W_{l,2k+1}$ ($l \geq 1$, $i \geq 0$). The significant advantage of this approach is that we can still easily update the AWSOM equations online, as new data values arrive. This is possible because the equation is selected based only on l and t for the new wavelet coefficient (see also Figure 3).

Definition 3. *For the full model described above, we use the notation*

$$\mathrm{AWSOM}_{\Lambda,T}(n_0, n_1, \ldots, n_\lambda)$$

The number of equations m is simply $m = \Lambda T$.

The value of Λ depends on N and the value of T is fixed and depends only on λ. In general, we *automatically* pick the following values:

$$T \sim 2^\lambda \quad \text{and} \quad \Lambda \sim \lg \frac{N}{N_\Lambda T} = \lg N - \lg N_\Lambda - \lambda$$

ModelSelection:
For each linear model:
Estimate SSR of complete model
For each subset of regression variables:
Compute SSR of reduced model from 3
Estimate probability that reduction in variance is not due to chance
Select the subset of variables with highest probability (or keep all if not within 95% confidence interval)

Figure 5: High-level description of basic selection algorithm.

4.3 Model selection

When fitting a linear model, as we increase the number of variables (or, in the context of forecasting, as we increase the window size), we expect in general to get a better fit. However, what we would really like to find are those variables that have a *statistically significant* contribution to the output value (or, forecast). The reasons for this are twofold:

- *Over*-fitting the data may result in a good approximation of the past, but a number of the correlations may in reality be due to noise and not carry over well in forecasts of the future. Therefore, by picking only the *important* variables, we improve accuracy.
- More importantly, in the pattern-mining context, we want to filter out the effects of noise and present only those patterns that are important to the user.

More details on model selection are given in Appendix D (see also Figure 5).

Model selection and combination needs to be done when interpreting the models; processing on the sensor is possible, but not necessary. In fact, all operations can be performed using *only* data gathered online and time complexity is *independent* of the stream size. The only thing that needs to be decided in advance is the largest $\mathrm{AWSOM}(n_0, \ldots, n_\lambda)$ order we may want to fit. From the data collected, we can then automatically select any model of smaller order ($\mathrm{AWSOM}(n'_0, \ldots, n'_{\lambda'})$, where $\lambda' \leq \lambda$ and $n'_i \leq n_i$).

4.4 Complexity

In this section we show that our proposed AWSOM models can be easily estimated with a single-pass, "any-time" algorithm. From Lemma 1, estimating the new wavelet coefficients requires space $O(\lg N)$. In fact, since we typically use Daubechies-6 wavelets ($L = 6$), we need to keep exactly $5 \lg N + 6$ values. The AWSOM models can be dynamically updated using RLS.

Lemma 2 (Logarithmic space complexity). *Maintaining the model requires $O(\lg N + mk^2)$ space, where N is the length of the signal so far, k is the number of* AWSOM *coefficients in each equation and m the number of equations.*

Dataset	Size	Description
`Triangle`	64K	Triangle wave (period 256)
`Mix`	256K	Square wave (period 256) plus sine (period 64)
`Impulses`	64K	Impulse train (every 256 points)
`ARFIMA`	2K	Fractionally differenced ARIMA (R package `fracdiff`).
`Sunspot`	2K	Sunspot data
`Disk`	2K	Disk access trace (from Hewlett-Packard)
`Automobile`	32K	Automobile traffic sensor trace from large Midwestern state

Table 3: Description of datasets (sizes are in number of points, 1K=1024 points).

Proof. Keeping the wavelet crest scaling coefficients requires space $O(\lg N)$. If we use recursive least squares, we need to maintain a $k \times k$ matrix for each of the m equations in the model. □

Auto-regressive models with a comparable window size need space $O(m^2 k^2)$, since the equivalent fair window size is $W \approx mk$. Here, "fair" means that the number of total number of AWSOM coefficients plus the number of initial conditions we need to store is the same for both methods. This is the information that comprises the data synopsis and that would have to be eventually communicated. However, the device gathering the measurements needs extra storage space in order to update the models. The latter is, in fact, *much* larger for AR than for AWSOM (see Figure 6). Thus this definition of equivalent window actually favors AR.

Theorem 1 (Time complexity). *Updating the model when a new data point arrives requires $O(k^2)$ time on average, where k is the number of* AWSOM *coefficients in each equation.*

Proof. On average, the wavelet crest scaling coefficients can be updated in $O(1)$ amortized time. Although a single step may require $O(\lg N)$ time in the worst case, on average, the (amortized) time required is $O(\sum_{i=0}^{n} \mathcal{B}(i)/N) = O(1)$ (where $\mathcal{B}(i)$ is the number of trailing zeros in the binary representation of i)[1].

Updating the $k \times k$ matrix for the appropriate linear equation (which can be identified in O(1), based on level l and on $t \mod T$), requires time $O(k^2)$. □

Once again, auto-regressive models with a comparable window size need time $O(m^2 k^2)$ for each update.

Corollary 1 (Constant-time update). *When the model parameters have been fixed (typically k is a small constant ≈ 10 and $m \sim \lg N$), the model requires space $O(\lg N)$ and amortized time $O(1)$ for each update.*

[1] Seen differently, *IncDWT* is essentially an pre-order traversal of the wavelet coefficient tree.

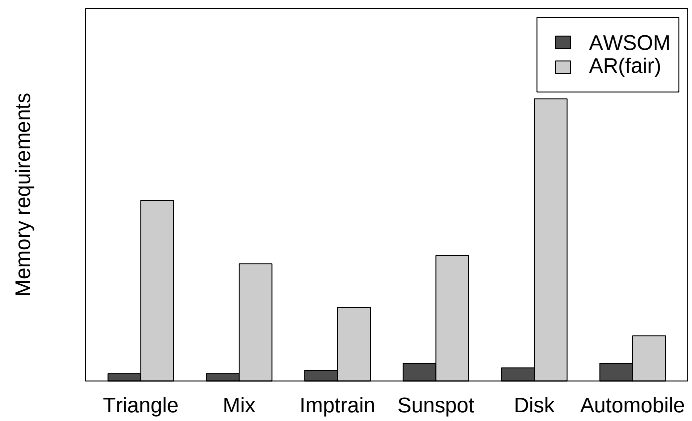

Figure 6: Memory space requirements: Space needed to keep the models up-to-date (AWSOM and AR with equivalent, fair window size.

5 Experimental evaluation

We compared AWSOM against standard AR (with the equivalent, fair window size—see Section 4.4), as well as hand-tuned (S)ARIMA (wherever possible). Our prototype AWSOM implementation is written in Python, using Numeric Python for fast array manipulation. We used the standard `ts` package from R (version 1.6.0—see `http://www.r-project.org/`) for AR and (S)ARIMA models. We illustrate the properties of AWSOM and how to interpret the models using synthetic datasets and then show how these apply to real datasets (see Table 3).

Only the first half of each sequence was used to estimate the models, which were then applied to generate a sequence of length equal to that of the *entire* second half. For AR and (S)ARIMA, the last values (as dictated by the window size) of the first half were used to initiate generation. For AWSOM we again used as many of the last wavelet coefficients from each DWT level of the first half as were necessary to start applying the model equations. We should note that generating more than, say, 10 steps ahead is very rare: most methods in the literature [WG94] generate one step ahead, then obtain the correct value of X_{t+1}, and *only then* try to generate X_{t+2}. Nevertheless, our goal is to capture long-term behavior and AWSOM achieves this efficiently, unlike ARIMA.

5.1 Interpreting the models

Visual inspection A "forecast" is essentially a by-product of any *generative* time series model: application of any model to generate a number of "future" values reveals precisely the trends and patterns captured by that model. In other words, synthesizing points based on the model is the simplest way for any user to get a quick, yet fairly accurate idea of what the trends are or, more precisely, what the *model* thinks they are. Thus, what we expect to see (especially in a long-range forecast) is the *important* patterns that can be identified from the real data.

However, an expert user can extract even more precise information from the models. We will now

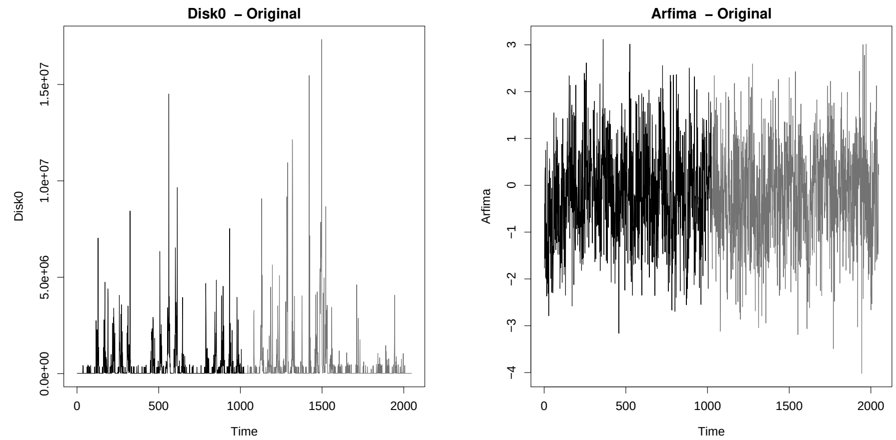

Figure 7: `Disk` and `ARFIMA` datasets.

explain how the "AWSOM language" can be fully interpreted.

Variance test As explained in Appendix B.1, if the signal is self-similar, then the plot of log-power versus scale is linear.

Definition 4 (Variance diagnostic). *We call the log-power vs. scale plot the* wavelet variance diagnostic plot *(or just* variance diagnostic*). In particular, we use the correlation coefficient ρ_α to quantify the relation. If the plot is linear (in a range of scales), the slope $\hat{\alpha}$ is the* self-similarity exponent *($-1 < \alpha < 0$, closer to zero the more bursty the series).*

A large value of $|\rho_\alpha|$, at least across several scales, indicates that it the series component in those scales may be modeled using a fractional noise process with parameter dictated by α (see `Automobile` dataset). However, we should otherwise be careful in drawing further conclusions about the behavior within these scales.

We should note that after the observation by [LTWW94], fractional noise processes and, in general, self-similar sequences have revolutionized network traffic modeling. Furthermore, self-similar sequences appear in atomic clock fluctuations, river minima, compressed video bit-rates [Ber94, PW00], to mention a few examples.

Wavelet variance (energy and power) The magnitude of variance within each scale serves as an indicator about which frequency components are the dominant ones in the sequence. To precisely interpret the results, we also need to take into account the fundamental uncertainty in frequencies (see Figure 15). However, the wavelet variance plot quickly gives us the general picture of important trends. Furthermore, it guides us to focus on AWSOM coefficients around frequencies with large variance.

AWSOM **coefficients** Regardless of the energy within a scale, the AWSOM coefficients provide further information about the presence of trends in the signal, which cannot be deduced from the variance plots. In particular:

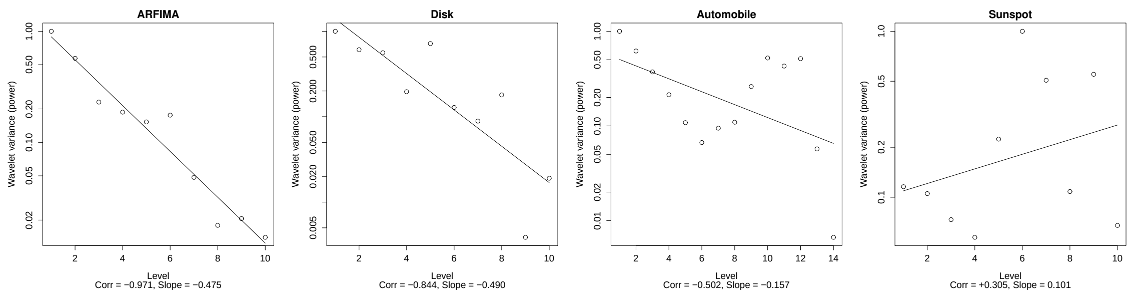

Figure 8: Wavelet variance diagnostic.

- Large intra-scale coefficients: These capture patterns at certain frequencies, regardless of the contribution of these frequencies to overall energy. Furthermore, if the coefficients are not the same for all regression models at the same level, this is an indication of "seasonalities" within that scale and capture a different type of information about larger frequencies.

- Large inter-scale coefficients: These occur when there are repeated bursts in the series. The number of scales with large inter-scale coefficients depends on the duration of the bursts (short bursts have large bandwidth).

To summarize, the steps are:

- Examine the variance diagnostic to identify sub-bands that correspond to a self-similar component. These may be modeled using a fractional noise process, but otherwise we cannot say much more.

- Examine the wavelet and energy and power spectrum to quickly identify important sub-bands.

- Examine AWSOM coefficients, primarily within and around the sub-bands identified during the second step.

5.2 Synthetic datasets

We present synthetic datasets to illustrate the basic properties of AWSOM, its behavior on several characteristic classes of sequences, and the principles behind interpreting the models. Applying the models to generate a number of "future" data points is the quickest way to see if each method captures long-term patterns.

`ARFIMA`. This is a synthetic dataset of a fractional noise process, which illustrates the first point about AWSOM model interpretation. As seen in Figure 8, this exhibits a clearly linear relationship in the wavelet variance plot. The estimated fractional differencing ("burstiness") parameter $\hat{\delta} \equiv -\hat{\alpha}/2 \approx 0.24$ is close to the actual parameter used ($\delta = 0.3$). It is clear that no periodic component is present in this series.

`Triangle`. AR fails to capture anything, because the window is not large enough. SAR estimation (with no differencing, no MA component and only a manually pre-specified lag-256 seasonal

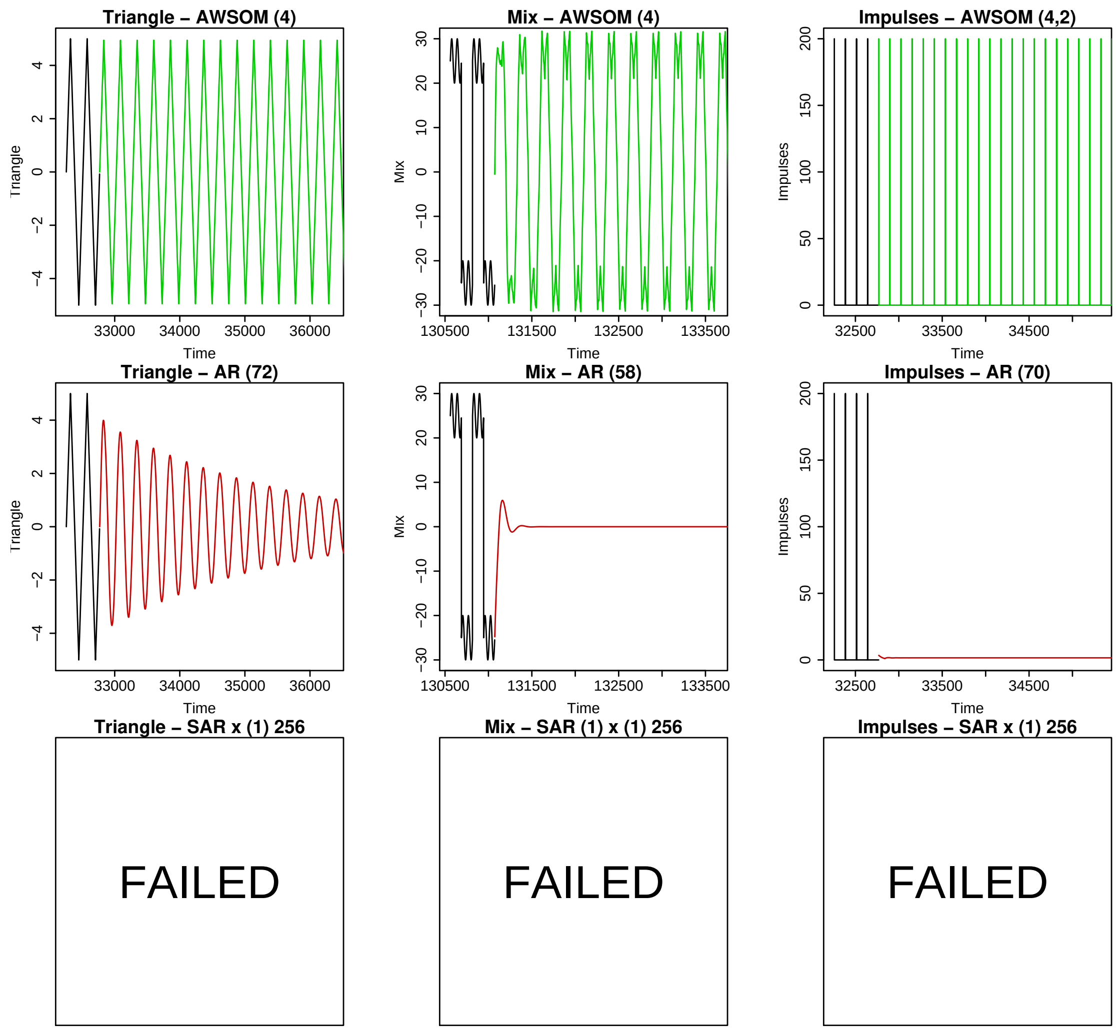

Figure 9: Forecasts—synthetic datasets. Note that AR gives the wrong trend (if any), while seasonal AR fails to complete.

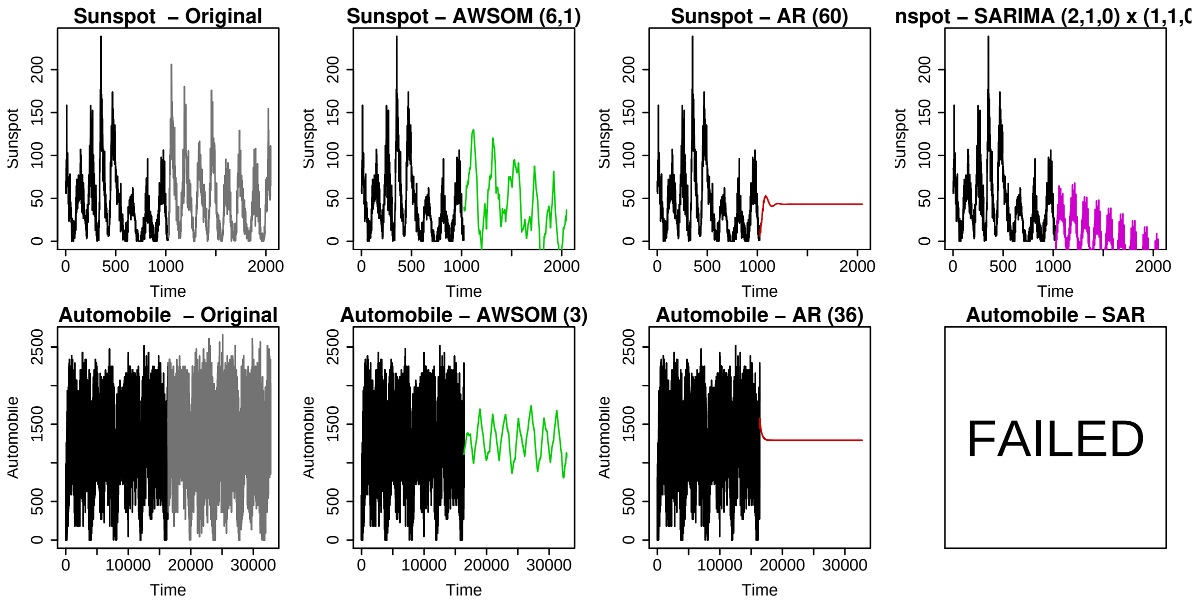

Figure 10: Forecasts—real datasets. Note that AR fails to detect any trend, while seasonal AR either fails to complete or gives a wrong conclusion (decaying trend, fixed period) in $260\times$ time.

component) fails completely. In fact, R segfaults after several minutes, even without using maximum-likelihood estimation (MLE). However, AWSOM captures the periodicity. The AWSOM model visualization is similar to that for `Mix`.

`Mix`. Once again, AR is confused and does not capture even the sinusoidal component. SAR estimation (without MLE) fails (R's optimizer returns an error, after several minutes of computation). Figure 14 shows the AWSOM coefficients. We show only the levels that correspond to significant variance. These illustrate the first point in the interpretation of AWSOM coefficients. We clearly see strong correlations in levels 6 and 8 (which correspond to the periods $2^6 = 64$ and $2^8 = 256$ of the series components). Note that the variance alone (see also Figure 15) is not enough to convey this information.

`Impulses`. Once again, AR fails to capture anything and SAR estimation fails. AR fails because the window is too small. However, AWSOM captures the overall behavior. Figure 14 illustrates the second point in the interpretation of AWSOM coefficients. We clearly see repeated presence of bursts, with strong inter-scale correlations across *all* levels up to the impulse "period" (since the bursts have width one). We show those levels that correspond to the bursts. At level 5, information from the impulse "period" begins to enter in the wavelet coefficients (see also Figure 15). After level 7, the inter-scale correlations diminish in significance and the interpretation is similar to that for `Mix`.

We should mention that we tried SAR$(0)\times(1)128$ on an impulse train of period 128. On a sequence with 1024 points, R takes over 4 minutes (on a signal with 64K points it did not complete in over one hour). However, AWSOM estimates the parameters (with 64K points) in approximately 50 seconds, although our prototype is implemented in Python.

5.3 Real datasets

For the real datasets, we show the so-called marginal distribution *quantile-quantile plots* (or *Q-Q plots*—see Figure 12 and Figure 11). These are the scatter plots of (x, y) such that $p\%$ of the values are below x in the real sequence and below y in the generated sequence. When the distributions are identical, the Q-Q plot coincides with the bisector of the first quadrant.

`Sunspot.` This dataset is well-known and it has a time-varying period. AR again fails completely. SAR (without a MA component, much less MLE) takes 40 minutes to estimate. AWSOM (in Python) takes less than 9 seconds. SAR gives a completely fixed period, captures a false downward trend and misses the marginal distribution (see Figure 12). On the other hand, AWSOM captures the general periodic trend, with a desirable slight confusion about the period (since the period is varying and thus un-predictable).

`Automobile.` This dataset has a strongly linear variance diagnostic in scales 1–6 (Figure 8). However, the lower frequencies contain the most energy, as can be seen in the variance plot (Figure 13. This is an indication that we should focus at these scales. The lowest frequency corresponds to a daily periodicity (approximately 4000 points per day, or about 8 periods in the entire series) and next highest frequency corresponds to the morning and afternoon rush-hour.

In this series, low frequencies can be modeled by fractional noise. Figure 11 shows a generated sequence with fractional noise, as identified by AWSOM. The fractional difference parameter is estimated as $\hat{\delta} \equiv -\hat{\alpha}/2 \approx 0.276$ and the amplitude is chosen to match the total variance in those scales.

However, for unsupervised outlier detection, this is not necessary: what would really constitute an outlier would be, for instance, days that (a) do not follow the daily and rush-hour patterns, or (b) whose variance in the fractional noise scales is very different. This can be captured automatically by the series components in the appropriate frequency sub-bands that AWSOM identifies as a periodic component and self-similar noise, respectively.

`Disk.` This is a very difficult data set to characterize (even by humans). It exhibits a linear variance diagnostic (see Figure 8) across *all* scales. Therefore, the regression models are of little use in this case. However, from the variance plot, we see moderate spikes at frequencies that correspond to daily periodicities (48 points per day) and, to a lesser extent, weekly periodicities. However, the presence in those frequencies is fairly week and the points (2K) are too few to *safely* draw any further

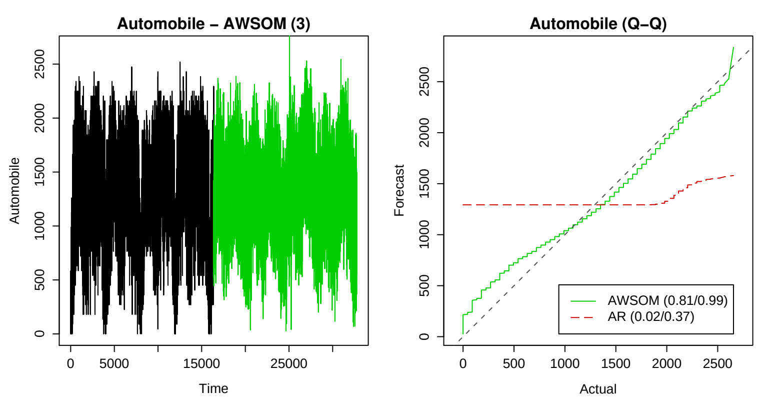

Figure 11: `Automobile`—forecast with fractional noise.

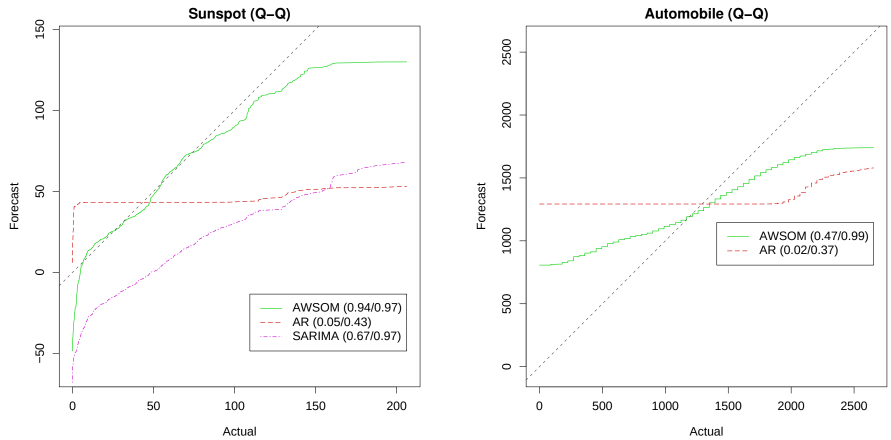

Figure 12: Marginal Q-Q plots (slope and correlation coefficients in parentheses).

conclusions; AWSOM provides the necessary information to judge what conclusions can be drawn. Both AR and SAR (with a hand-picked lag of 48 to capture the daily periods) fail completely and do not even provide a hint about the weak (compared to the bursts) periodic components.

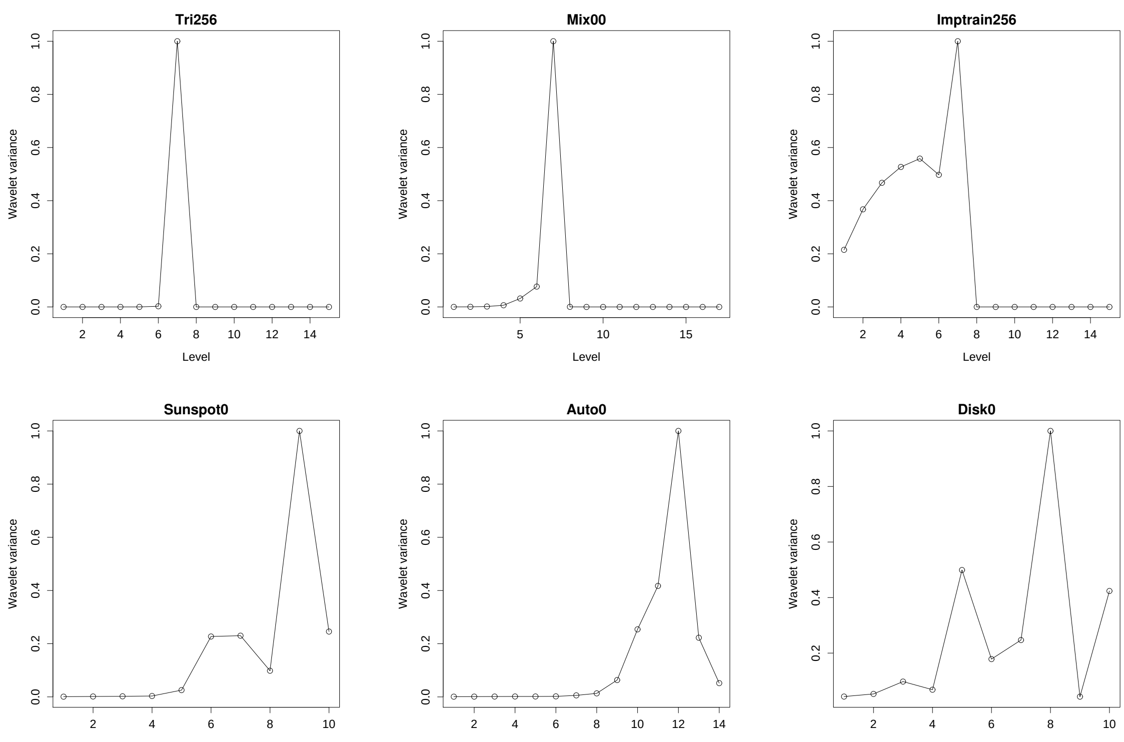

Figure 13: Wavelet variances.

6 Conclusions

Sensor networks are becoming increasingly popular, thanks to falling prices and increasing storage and processing power. In this work we presented AWSOM (Arbitrary-Window Stream mOdeling Method). Our method is the only one that achieves all our goals: *(1) Unsupervised operation:* once we decide the largest AWSOM order we want, no further intervention is needed: the sensor can be left alone to collect information. *(2) 'Any-time', one-pass* algorithm to incrementally update the patterns. *(3) Automatic* detection of periodic components with arbitrary period. *(4) Limited memory:* our method requires $O(\lg N)$ memory (where N is the length of the sequence so far). *(5) Simplicity:* AWSOM provides linear models which have a straightforward interpretation. *(6) Power:* AWSOM provides information across several frequencies and can diagnose self-similarity and long-range dependence. *(7) Immediate outlier detection:* our method, despite its simplicity and its automatic, unsupervised operation, is nevertheless able to do forecasting. It can do so directly, for the estimated model. We showed real and synthetic data, where our method captures the periodicities and burstiness of the input sequence, while the traditional ARIMA method fails completely.

AWSOM is an important first step toward successful, hands-off data mining in infinite streams, combining simplicity with modeling power. Continuous queries are useful for evidence gathering and hypothesis testing *once* we know what we are looking for. AWSOM is the first method to deal directly with the problem of automatic, unsupervised stream mining and pattern detection and fill the gap. Among the many future research directions, the most promising seems to be the extension of AWSOM to multiple, co-evolving time sequences, beyond MUSCLES [YSJ+00] and multi-variate ARIMA.

Acknowledgments We would like to thank Becky Buchheit for her valuable help with the automobile traffic datasets.

References

[ABB+02] Arvind Arasu, Brian Babcock, Shivnath Babu, Jon McAlister, and Jennifer Widom. Characterizing memory requirements for queries over continuous data streams. In *Proc. 21st PODS*, 2002.

[Aka97] Metin Akay, editor. *Time Frequency and Wavelets in Biomedical Signal Processing*. John Wiley and Sons, 1997.

[BD91] Peter J. Brockwell and Richard A. Davis. *Time Series: Theory and Methods*. Springer Series in Statistics. Springer-Verlag, 2nd edition, 1991.

[Ber94] Jean Beran. *Statistics for Long-Memory Processes*. Chapman & Hall, 1994.

[BGS01] Philippe Bonnet, Johannes E. Gehrke, and Praveen Seshadri. Towards sensor database systems. In *Proc. 2nd MDM*, pages 3–14, 2001.

[BJR94] George E.P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, 3rd edition, 1994.

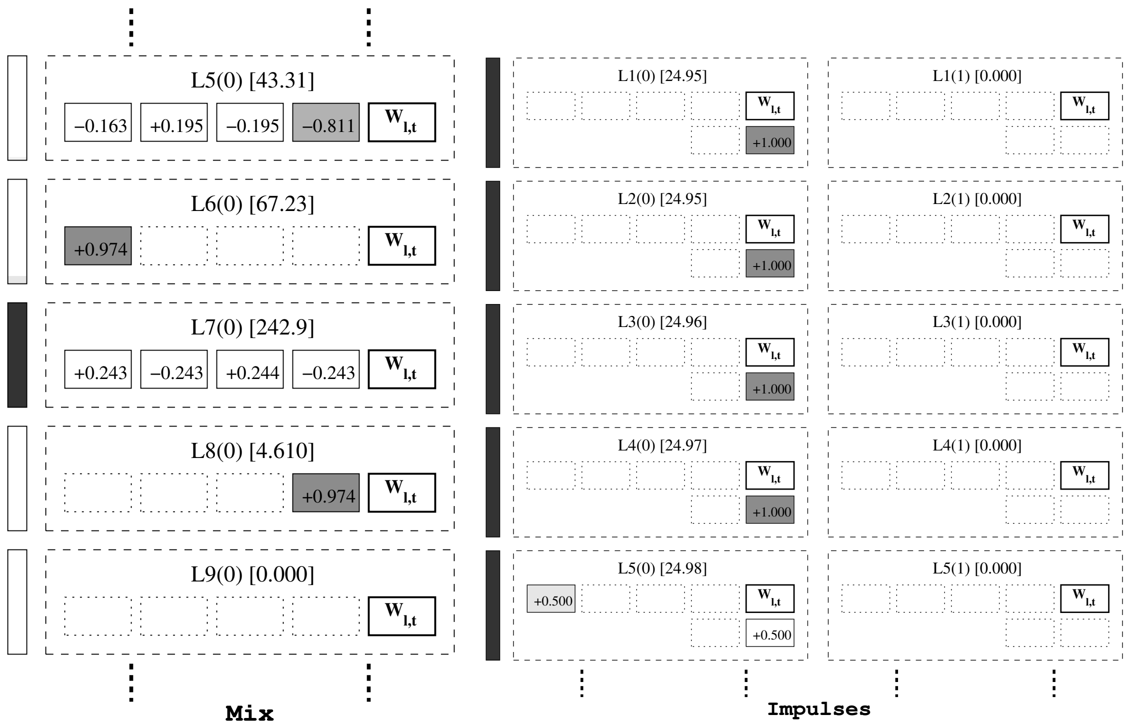

Figure 14: `Mix` and `Impulses`—AWSOM models. The bars on the left are proportional to the variance (see also Figure 13) and the numbers in brackets show the variance of the wavelet coefficients that correspond to each AWSOM equation. Blank boxes correspond to coefficients excluded by the model selection process.

[Bol86] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31:307–327, 1986.

[CB96] M. Crovella and A. Bestavros. Self-similarity in world wide web traffic, evidence and possible causes. *SIGMETRICS*, pages 160–169, 1996.

[CBF+00] L.R̃ichard Carley, James A. Bain, Garry K. Fedder, David W. Greve, David F. Guillou, Michael S.C. Lu, Tamal Mukherjee, Suresh Santhanam, Leon Abelmann, and Seungook Min. Single-chip computers with microelectromechanical systems-based magnetic memory (invited). *J. Applied Physics*, 87(9):6680–6685, 2000.

[CCC+02] Donald Carney, Ugur Cetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Greg Seidman, Michael Stonebraker, Nesime Tatbul, and Stanley B. Zdonik. Monitoring streams – a new class of data management applications. In *Proc. 28th VLDB*, 2002.

[CDH+02] Yixin Chen, Guozhu Dong, Jiawei Han, Benjamin W. Wah, and Jianyong Wang. Multidimensional regression analysis of time-series data streams. In *Proc. 28th VLDB*, 2002.

[CGN00] L.R̃ichard Carley, Gregory R. Ganger, and David Nagle. Mems-based integrated-circuit mass-storage systems. *CACM*, 43(11):72–80, 2000.

[DGGR02] Alin Dobra, Minos N. Garofalakis, Johannes Gehrke, and Rajeev Rastogi. Processing complex aggregate queries over data streams. In *Proc. SIGMOD*, 2002.

[DGI+02] Mayur Datar, Aristides Gionis, Piotr Indyk, , and Rajeev Motwani. Maintaining stream statistics over sliding windows. In *Proc. 13th SODA*, 2002.

[Fal96] Christos Faloutsos. *Searching Multimedia Databases by Content*. Kluwer Academic Inc., 1996.

[Fie93] D.J. Field. Scale-invariance and self-similar 'wavelet' transforms: An analysis of natural scenes and mammalian visual systems. In M. Farge, J.C.R. Hunt, and J.C. Vassilicos, editors, *Wavelets, Fractals, and Fourier Transforms*, pages 151–193. Clarendon Press, Oxford, 1993.

[GG02] Minos N. Garofalakis and Phillip B. Gibbons. Wavelet synopses with error guarantees. In *Proc. SIGMOD*, 2002.

[GGI+02a] Anna C. Gilbert, Sudipto Guha, Piotr Indyk, Yannis Kotidis, S. Muthukrishnan, and Martin J. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proc. 34th STOC*, pages 389–398, 2002.

[GGI+02b] Anna C. Gilbert, Sudipto Guha, Piotr Indyk, S. Muthukrishnan, and Martin Strauss. Near-optimal sparse fourier representations via sampling. In *Proc. 34th STOC*, pages 152–161, 2002.

[GKMS01] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proc. 27th VLDB*, pages 79–88, 2001.

[GKS01] Johannes Gehrke, Flip Korn, and Divesh Srivastava. On computing correlated aggregates over continual data streams. In *Proc. SIGMOD*, 2001.

[GKS02] Sudipto Guha, Nick Koudas, and Divesh Srivastava. Fast algorithms for hierarchical range histogram construction. In *Proc. PODS*, pages 180–187, 2002.

[GSW01] Ramazan Gencay, Faruk Selcuk, and Brandon Whitcher. *An Introduction to Wavelets and Other Filtering Methods in Finance and Economics*. Academic Press, 2001.

[HSW+00] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. In *Proc. ASPLOS-IX*, 2000.

[IKM00] Piotr Indyk, Nick Koudas, and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In *Proc. 26th VLDB*, pages 363–372, 2000.

[JL84] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[LTWW94] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. On the self-similar nature of ethernet traffic. *IEEE Trans. on Networking*, 2(1):1–15, 1994.

[MSHR02] Samuel R. Madden, Mehul A. Shah, Joseph M. Hellerstein, and Vijayshankar Raman. Continuously adaptive continuous queries over streams. In *SIGMOD Conf.*, 2002.

[OS75] Alan Victor Oppenheim and Ronald W. Schafer. *Digital Signal Processing*. Prentice-Hall, 1975.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.

[PW00] Donald B. Percival and Andrew T. Walden. *Wavelet Methods for Time Series Analysis*. Cambridge University Press, 2000.

[RFGN00] Erik Riedel, Christos Faloutsos, Gregory R. Ganger, and David Nagle. Data mining on an OLTP system (nearly) for free. In *SIGMOD Conf.*, pages 13–21, 2000.

[RMSCB99] Rudolf H. Riedi, Vinay J.R̃ibeiro Matthew S.C̃rouse, and Richard G. Baraniuk. A multifractal wavelet model with application to network traffic. *IEEE Special Issue on Information Theory*, 45:992–1018, 1999.

[SGNG00] Steven W. Schlosser, John Linwood Griffin, David F. Nagle, and Gregory R. Ganger. Designing computer systems with mems-based storage. In *Proc. ASPLOS-IX*, 2000.

[WG94] Andreas S. Weigend and Neil A. Gerschenfeld. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison Wesley, 1994.

[WMC+02] Mengzhi Wang, Tara M. Madhyastha, Ngai Hang Chan, Spiros Papadimitriou, and Christos Faloutsos. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. In *Proc. 18th ICDE*, 2002.

[You84] Peter Young. *Recursive Estimation and Time-Series Analysis: An Introduction*. Springer-Verlag, 1984.

[YSJ+00] Byoung-Kee Yi, N.D. Sidiropoulos, Theodore Johnson, H.V. Jagadish, Christos Faloutsos, and Alexandros Biliris. Online data mining for co-evolving time sequences. *Proc. 16th ICDE*, pages 13–22, 2000.

[ZdZ98] R.A. Zuidwijk and P.M. de Zeeuw. Fast algorithm for directional time-scale analysis using wavelets. In *Proc. of SPIE, Wavelet Applications in Signal and Imaging Processing VI*, volume 3458, pages 222–231, 1998.

A Auto-regressive modeling

The simplest form, an auto-regressive model of order p or $\mathrm{AR}(p)$, expresses X_t as a linear combination of previous values, i.e., $X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \epsilon_t$ or, more concisely

$$\phi(L)X_t = \epsilon_t$$

where L is the lag operator and $\phi(L)$ is a polynomial defined on this operator:

$$
\begin{aligned}
LX_t &\equiv X_{t-1} \\
\phi(L) &= 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p
\end{aligned}
$$

and ϵ_t is a white noise process, i.e.,

$$
\mathrm{E}[\epsilon_t] = 0 \quad \text{and} \quad \mathrm{Cov}[\epsilon_t, \epsilon_{t-k}] = \begin{cases} \sigma^2 & \text{if } k = 0 \\ 0 & \text{otherwise} \end{cases}
$$

Using least-squares, we can estimate σ^2 from the sum of squared residuals (SSR). This is used as a measure of estimation error; when generating "future" points, ϵ_t is set to its expected value, $\mathrm{E}[\epsilon_t] \equiv 0$.

The next step up are auto-regressive moving average models. An $\mathrm{ARMA}(p, q)$ model expresses values X_t as

$$
\phi(L)X_t = \theta(L)\epsilon_t
$$

where $\theta(L) = 1 - \theta_1 L - \cdots - \theta_q L^q$. Estimating the moving average coefficients θ_i is fairly involved. State of the art methods use maximum-likelihood (ML) algorithms, employing iterative methods for non-linear optimization, whose computational complexity depends exponentially on q.

$\mathrm{ARIMA}(p, d, q)$ models are similar to $\mathrm{ARMA}(p, q)$ models, but operate on $(1 - L)^d X_t$, i.e., the d-th order backward difference of X_t:

$$
\phi(L)(1 - L)^d X_t = \theta(L)\epsilon_t
$$

Finally, $\mathrm{SARIMA}(p, d, q) \times (P, D, Q)T$ models are used to deal with seasonalities, where:

$$
\phi(L)\Phi(L^T)(1 - L)^d(1 - L^T)^D X_t = \theta(L)\Theta(L^T)\epsilon_t
$$

where the seasonal difference polynomials

$$
\begin{aligned}
\Phi(L^T) &= 1 - \Phi_1 L^T - \Phi_2 L^{2T} - \cdots - \Phi_P L^{PT} \\
\Theta(L^T) &= 1 - \Theta_1 L^T - \Theta_2 L^{2T} - \cdots - \Theta_Q L^{QT}
\end{aligned}
$$

are similar to $\phi(L)$ and $\theta(L)$ but operate on lags that are multiples of a fixed period T. The value of T is determined purely by trial and error, or by utilizing prior knowledge about the series X_t.

B Discrete Wavelet Transform

Wavelets are best introduced with the Haar transform, because of its simplicity. At each level l of the construction we keep track of two sets of coefficients, each of which "looks" at a time window of size 2^l:

- $W_{l,t}$: The *smooth* component, which consists of the $N/2^l$ *scaling coefficients*. These capture the low-frequency component of the signal; in particular, the frequency range $[0, 1/2^l]$.
- $V_{l,t}$: The *detail* component, which consists of the $N/2^l$ *wavelet coefficients*. These capture the high-frequency component; in particular, the frequency range $[1/2^l, 1/2^{l-1}]$.

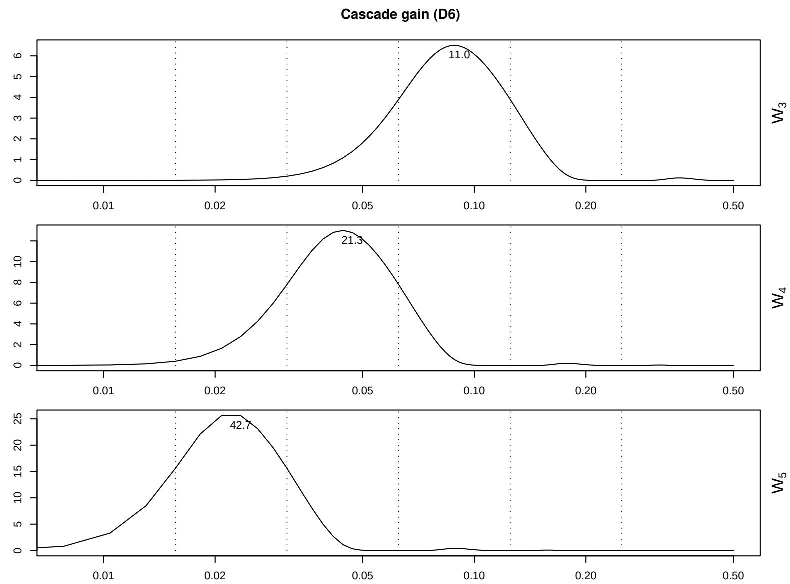

Figure 15: Daubechies-6 cascade gain (levels 3–5).

The construction starts with $V_{0,t} = X_t$ and $W_{0,t}$ is not defined. At each iteration $l = 1, 2, \ldots, \lg N$ we perform two operations on $V_{l-1,t}$ to compute the coefficients at the next level:

- Differencing, to extract the high frequencies:

$$W_{l,t} = (V_{l-1,2t} - V_{l-1,2t-1})/\sqrt{2}$$

- Smoothing, by averaging[2] each consecutive pair of values to extract the low frequencies:

$$V_{l,t} = (V_{l-1,2t} + V_{l-1,2t-1})/\sqrt{2}$$

We stop when $W_{l,t}$ consists of one coefficient (which happens at $l = \lg N+1$). The scaling coefficients are needed only during the intermediate stages of the computation. The final wavelet transform is the set of all wavelet coefficients along with $V_{\lg N+1,0}$. Starting with $V_{\lg N+1,0}$ (which is also referred to as the signal's scaling coefficient) and following the inverse steps, we can reconstruct each $V_{l,t}$ until we reach $V_{0,t} \equiv X_t$.

Figure 1 illustrates the final effect for a signal with $N = 16$ values. Each wavelet coefficient is the result of projecting the original signal onto the corresponding basis signal.

In general, there are many wavelet transforms, but they all follow the pattern above: a wavelet transform uses a pair of filters, one high-pass and one low-pass. In the case of the Haar transform Different wavelet families (e.g., Coiflets, least-asymmetric, to mention a few) achieve different trade-offs with respect to (un)smoothness of the projections, phase shift properties, etc [PW00].

[2]The scaling factor of $1/\sqrt{2}$ in both the difference and averaging operations is present in order to preserve total signal energy (i.e., sum of squares of all values).

Frequency properties Wavelet filters employed in practice can only approximate an ideal bandpass filter, since they are of *finite* length L. This is an unavoidable consequence of the uncertainty principle. The practical implications are that wavelet coefficients at level l correspond roughly to frequencies $[1/2^{l+1}, 1/2^l]$ (or, equivalently, periods $[2^l, 2^{l+1}]$ (see Table 4 and Figure 15 for the actual correspondence). This has to be taken into account for *precise* interpretation of AWSOM models.

B.1 Wavelet variance and self-similarity

The wavelet variance decomposes the variance of a sequence across scales. Here we mention the definitions and basic facts; details can be found in [PW00].

Definition 5 (Wavelet variance). *If $\{W_{l,t}\}$ is the DWT of a series $\{X_t\}$ then the wavelet variance $\mathcal{V}_l$ is defined as*

$$\mathcal{V}_l = \mathrm{Var}[W_{l,t}]$$

Under certain general conditions

$$\hat{\mathcal{V}}_l = \frac{2^l}{N} \sum_{t=1}^{N/2^l} W_{l,t}$$

is an unbiased estimator of $\mathcal{V}_l$. Note that the sum is precisely the energy of $\{X_t\}$ at scale l.

Definition 6 (Self-similar sequence). *A sequence $\{X_t\}$ is said to be self-similar following a pure power-law process if*

$$\mathcal{S}_X(f) \propto |f|^\alpha$$

where $-1 < \alpha < 0$ and $\mathcal{S}_X(f)$ is the SDF[3]

It can be shown that

$$\mathcal{V}_l \approx 2 \int_{1/2^{l+1}}^{1/2^l} \mathcal{S}_X(f) df$$

thus if $\{X_t\}$ is self-similar, then

$$\log \mathcal{V}_l \propto l$$

i.e., the plot of $\log \mathcal{V}_l$ versus the level l should be linear. In fact, slope of the log-power versus scale plot should be approximately equal to the exponent α. This fact and how to estimate $\mathcal{V}_l$ are what the reader needs to keep in mind.

[3]The *spectral density function (SDF)* is the Fourier transform of the auto-covariance sequence (ACVS) $S_{X,k} \equiv \mathrm{Cov}[X_t, X_{t-k}]$. Intuitively, it decomposes the variance into frequencies.

	Periods			
	Ideal	Non-zero	Dominant	Peak
4	16–32	11–45	14–28	23
5	32–64	23–109	29–57	45
6	64–128	41–205	58–111	91
7	128–256	157–440	111–212	181

Table 4: Frequency information content of Daubechies-6 wavelet coefficients per level.

C Recursive Least Squares (RLS)

Let us assume that X is an $m \times k$ matrix of m measurements (one set of k input variables per row), $\mathbf{b}$ is the $k \times 1$ vector of regression coefficients and $\mathbf{y}$ the $m \times 1$ vector of outputs. The LS solution to the overdetermined system $X\mathbf{b} = \mathbf{y}$ is the solution of

$$X^T X\mathbf{b} = X^T\mathbf{y} \tag{4}$$

When a new vector $\mathbf{x}_{m+1}$ and output y_{m+1} arrive, we can update the $k \times k$ projection matrix $X^T X$ by adding the outer product $\mathbf{x}_{m+1}\mathbf{x}_{m+1}^T$ to it. Similarly, we can update $X^T\mathbf{y}$ by adding $y_{m+1}\mathbf{x}_{m+1}$. Since all we need for the solution are

$$P \equiv X^T X \qquad \text{and} \qquad \mathbf{q} \equiv X^T\mathbf{y}$$

we need only space $O(k^2 + k) = O(k^2)$ to keep the model up to date. In fact, it is possible to update the regression coefficient vector $\mathbf{b}$ at each step without explicitly solving Equation 4 (see [You84]).

D Model selection

We show how feature selection with model combination can be done from the data gathered online (i.e., P and $\mathbf{q}$ for each AWSOM equation).

D.1 Model testing

Lemma 3 (Square sum of residuals). *If* $\mathbf{b}$ *is the least-squares solution to the overdetermined equation* $X\mathbf{b} = \mathbf{y}$*, then*

$$s_n \equiv \sum_{i=1}^{n} (\mathbf{x_i}^T\mathbf{b} - y_i)^2 = \mathbf{b}^T P\mathbf{b} - 2\mathbf{b}^T\mathbf{q} + \mathbf{y}^2$$

Proof. Straightforward from the definition of s_n, which in matrix form is $s_n = (X\mathbf{b} - \mathbf{y})^2$. ☐

Thus, besides P and $\mathbf{q}$, we only need to update $\mathbf{y}^2$ (a single number), by adding y_i^2 to it as each new value arrives.

Now, if we select a subset $\mathcal{I} = \{i_1, i_2, \ldots, i_p\} \subseteq \{1, 2, \ldots, k\}$ of the k variables $x_1, x_2, \ldots, x_k$, then the solution $\mathbf{b}_\mathcal{I}$ for this subset is given by $P_\mathcal{I}\mathbf{b}_\mathcal{I} = \mathbf{q}_\mathcal{I}$ and the SSR by $s_n = \mathbf{b}_\mathcal{I}^T P_\mathcal{I}\mathbf{b}_\mathcal{I} - 2\mathbf{b}_\mathcal{I}\mathbf{q}_\mathcal{I} + \mathbf{y}^2$ where the subscript $\mathcal{I}$ denotes straight row/column selection (e.g., $P_\mathcal{I} = [p_{i_j,i_k}]_{i_j,i_k \in \mathcal{I}}$)

The *F-test (Fisher test)* is a standard method in statistics to determine whether a reduction in variance is statistically significant. In particular, if f is the ratio of the sample variance of the simplified model (i.e., one with fewer variables) to the sample variance of the complete model, then the *F-distribution* describes the probability that f takes a certain value, assuming that the difference is due to chance.

The F-test is based on the sample variances, which can be computed *directly* from the SSR, as explained in Lemma 3. The F-distribution holds precisely (i.e., non-asymptotically) under normality assumptions. However, in practice it works well in most circumstances, especially when the population size is large. This is clearly the case with semi-infinite streams.

D.2 Model combination

If we split measurements x_i into two subsets X_1 and X_2 with corresponding outputs $\mathbf{y}_1$ and $\mathbf{y}_2$, then the LS solution for both subsets combined is given by $b = (X^T X)^{-1} X^T y$ where $X = \left[X_1^T X_2^T\right]^T$ and $\mathbf{y} = [\mathbf{y}_1^T \mathbf{y}_2^T]^T$, i.e.,

$$b = (X_1^T X_1 + X_2^T X_2)^{-1}(X_1^T \mathbf{y}_1 + X_2^T \mathbf{y}_2) = (P_1 + P_2)^{-1}(q_1 + q_2)$$

Therefore, it is possible to combine sub-models when reducing the number of levels (effectively reducing the T parameter). Model selection as presented above can be extended to include this case.