SoftwareX ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Fast wavelet transform assisted predictors of streaming time series

Marco Stocchi [a,*], Michele Marchesi [b]

[a] Department of Electronics Engineering, University of Cagliari, v. Marengo 9, Italy
[b] Department of Mathematic Sciences and Informatics, University of Cagliari, v. Porcell 4, Italy

ARTICLE INFO

Article history:
Received 12 December 2016
Received in revised form 20 July 2017
Accepted 28 September 2017

Keywords:
Streaming datasets
Time series forecast
Fast wavelet transform
Shift variance theorem

ABSTRACT

We developed an implementation of a novel shift variance theorem of the fast wavelet transform (FWT), suitable to the multiresolution analysis of streaming univariate datasets, using compactly supported Daubechies Wavelets. The theorem is used to reduce the computational complexity of the FWT, and also to reduce drastically the number of wavelet coefficients to be estimated in forecasting the one step ahead discrete wavelet transform. For this reason, any FWT performed using the found shift variance properties is herein named reduced FWT. An effective real value prediction of a sampled input time series can be obtained performing the inverse DWT of an estimated crystal, and this is the purpose of the proposed predictor herein named Wa.R.P. (Wavelet transform Reduced Predictor). The C++ code implementing the FWT and the novel theorem is available to research purposes, and to build efficient industrial applications.

Code metadata

Current code version	v1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_SOFTX-D-17-00070
Legal Code License	MIT
Code versioning system used	none
Software code languages, tools, and services used	C++
Compilation requirements, operating environments & dependencies	TargetMachine x86, subsystem: Console. Compilation requirements and dependencies inherited from Microsoft Visual Studio 2015 project defaults for release x86 solution configuration
If available Link to developer documentation/manual	
Support email for questions	marco.stocchi@diee.unica.it

Software metadata

Current software version	1.0
Permanent link to executables of this version	
Legal Software License	MIT
Computing platforms/Operating Systems	Microsoft Windows (suggested)
Installation requirements & dependencies	only the source code is provided
If available, link to user manual — if formally published include a reference to the publication in the reference list	
Support email for questions	marco.stocchi@diee.unica.it

1. Motivation and significance

The use of wavelet analysis of time series has been demonstrated to be an effective and powerful method to analyze digital signals and, recently, it has been used also to forecasting purposes, as outlined in manuscript [1] sec. 1 "Introduction". In this paper,

* Corresponding author.
E-mail address: marco.stocchi@diee.unica.it (M. Stocchi).

https://doi.org/10.1016/j.softx.2017.09.006

we describe the main features of the proposed software, implementing a CPU efficient fast wavelet transform and its variant, a decimated DWT whose calculation is assisted by the novel shift variance theorem (SVT of the FWT), described in manuscript [1] sec. 2. "Method".

The majority of wavelet approaches to the analysis of digital signals involve the use of undecimated wavelet transforms, that are characterized by shift invariance features, but are also affected by representation redundancy and by a strong need of memory and computational resources. In manuscript [1] sec. 2 "Method" we demonstrate that the decimated, convolutional discrete wavelet transform (also called fast wavelet transform) possesses some important shifting features that are applicable to the multiresolution analysis (MRA) of streaming datasets, and that, under certain conditions, such features can be successfully used both to accelerate the computation of the FWT of streaming time series, and to efficiently preprocess univariate time series candidate to be forecasted via statistical estimators or machine learning predictors.

The proposed software implements both the FWT algorithm and its SVT variant. Moreover, motivated by the possibility to perform the FWT using the above mentioned theorem, we provide a test suitable to evaluate the theorem on the computational efficiency of the SVT assisted transform (also named reduced FWT).

The proposed software is endowed with a series of C++ files, created in order to let the user test separately various aspects of the contents illustrated in the manuscript [1].

The first test file (`T1.cpp`) allows to perform FWT operations on a streaming dataset, reporting the reconstruction errors for each different wavelet filter applied. The user can choose execution parameters such as the source size and the number of tests to be performed for each wavelet. Such test file also shows how to instantiate FWT transform objects at runtime.

The goal of the second test file (`T2.cpp`) is to test the FWT implementation and its SVT variant. The user can choose the number of correctness tests performed (these consist in comparing for equality the results obtained executing the classic FWT and the SVT assisted transform); it is also possible to choose the number of tests performed in order to evaluate the efficiency and, finally, the user could also repeat the tests changing the input source size. A final console report of the execution time allows the user to check the validity of the corollary on the asymptotic computational efficiency of the reduced FWT of streaming 1D datasets (see manuscript [1] sec. 2.1 "Theorem on coefficients transposition").

Finally, the case-study (the Bitcoin-USD hourly exchange rates prediction, i.e. a forecast attempt of a streaming financial time series) implements the Wa.R.P. engine (as proposed in the manuscript [1]). It is contained in a C++ file (named `DSPX_predictor.cpp`) which includes a set of header files containing support classes suitable to create a inference engine assisted by the reduced FWT; a full multilayered perceptron implementation (used for the machine learning purposes of the case study); a small set of files containing helper classes and functions suitable to manipulate financial series data. We also provide the files `DSPX_benchmark_ann.cpp`, `DSPX_benchmark_svm.cpp`, and `DSPX_benchmark_wdnn.cpp`, in order to respectively reproduce the forecasting benchmark results of a multilayered perceptron — MLP, support vector machines (the latter developed using the SVM implementation described in [2]), and Wavelet denoising-based neural networks — WDNN.

2. Software description

The FWT and the reduced FWT algorithms are developed using a template approach, and are declared in a C++ header file. An associated set of compilable files allow users and researchers to perform preliminary tests, and at the same time to learn how to use the algorithms by examples. The user should create a default C++ console project on its compiler and add the files listed in Appendix. The software must directly compile as a release x86 version (with optimization for speed) with no errors and no warnings. The user should include in the build both the project default `stdafx.cpp` and the `DSPX_singletons.cpp` files, but only one of the testing `.cpp` files listed in the table of Appendix under the section "Test".

2.1. Software architecture

The software, implementing the FWT and its SVT variant, is template based and developed in C++ language. Structures are gathered in a namespace contained in a single header file. This allows to maximize its reusability as a core component of both desktop and server applications.

2.2. Software functionalities

The code is composed of a structure implementing the shift variance theorem, an abstract structure implementing the fast wavelet transform interface, and a transformer class (derived from the abstract structure) that can be instantiated at compile time with the Daubechies orthonormal wavelet filters. A helper function is provided, useful to create Daubechies FWT transform objects at runtime.

The FWT abstract structure offers virtual methods to be called in order to perform the forward and inverse discrete wavelet transform using the FWT algorithm. It is endowed with two overloaded methods to be called in order to perform a forward transform. The first overload performs a classical FWT on the input series; the second overload allows to pass information about varying coefficients, back step sizes and the matrix Q, in order to perform a SVT transform, as described in manuscript [1] sec. 2.1 (Theorem on coefficients transposition).

The shift variance theorem structure is endowed with a constructor taking the input source size and the wavelet filter size as arguments. It features eight public methods, allowing the user to compute the needed parameters to perform a reduced FWT; the main ones are described in the followings.

The boolean method `is_scaling_coefficient()` allows to test if the ordinal index of a given coefficient of a transform crystal belongs to the ϕ coefficients of the DWT (the reader should refer to the coefficients indicated by eq. (8) of the manuscript [1], i.e.: $c_{M,0}, c_{M,1}, \ldots, c_{M,n_{c_M}}$).

The boolean method `is_SVT_coefficient()` detects if a coefficient, whose ordinal index is given, is one of the unchanging coefficients of the crystal (SVT coefficient). In this case, the coefficients can be automatically retrieved.

The method `back_steps()` allows to calculate, for a given SVT coefficient, the row of the Q matrix in which to fetch the retrievable coefficients. Such reassignment procedure is outlined in eq. (12) of the manuscript [1]. For efficiency purposes, the user is advised to cache the back steps in a vector, before performing a series of SVT transforms. A possible implementation of such procedure is provided in the `T2.cpp` test file (see `struct Q`, a wrapper object to work with a Q matrix).

Finally, the method `variant_coefficients()` allows to extract the number of non-SVT coefficients, as described in eq. (11) of the manuscript [1]. The interested reader can also look at Table A.1 of the manuscript, which reports the number of varying coefficients for each wavelet filter N and for each recursion step of the FWT of a streaming dataset.

Please cite this article in press as: Stocchi M., Marchesi M. Fast wavelet transform assisted predictors of streaming time series. SoftwareX (2017), https://doi.org/10.1016/j.softx.2017.09.006.

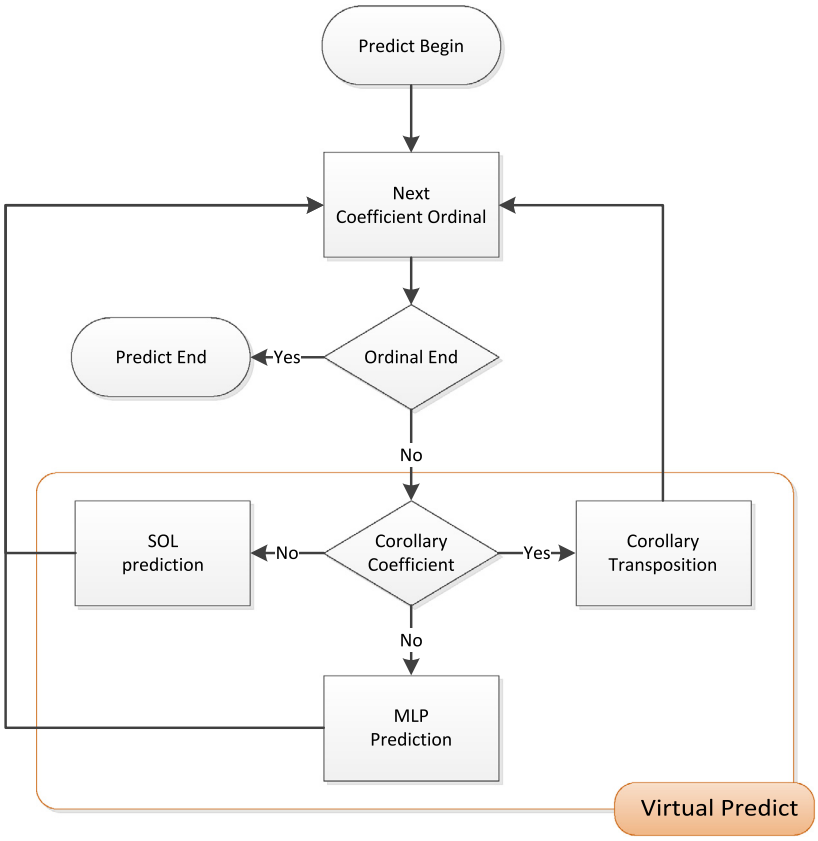

Fig. 1. Flow chart of a possible implementation of a Wa.R.P. system. At each forecast step, an iterator points to the index of the DWT coefficient to be forecasted. When the iterator points to a SVT coefficient, its actual value can be transposed from matrix Q. Otherwise, a specialized and trained Neural Network performs its prediction.

3. Illustrative examples

Fig. 1 shows the flowchart of a Wa.R.P. predictor system employing different types of machines for prediction purposes. it is similar to the software contained in the case study files. The representation reports a single forecasting step of the one step ahead crystal of a streaming dataset series. Theoretically, an iterator points to the coefficient ordinal index of the crystal to test the SVT theorem. If the iterator is effectively pointing to an SVT coefficient, there is no need to forecast the coefficient itself since it can be retrieved and transposed from matrix Q, as outlined in eq. (12) of the manuscript [1]. If the iterator points to a non SVT coefficient, the latter must be estimated and this can be performed testing a machine trained to such purpose.

In the header file `DSPX_engine.h`, belonging to the case study test, we parsimoniously provide only a set of multilayered perceptrons. However, far from being the only possible solution, a developer could decide to specialize the template predictor class with different types of predictors, depending on the operational requirements of the software. In the flowchart representation, for example, a theoretic virtual predictor can morph to different types of predictor machines (MLPs, Self Organizing Layers, SVMs, etc.); such decision could be taken considering the known statistical properties of the coefficients' series to forecast.

4. Impact

The software here proposed allows to explore and take advantage of the shift variance properties of the FWT which, at the best of our knowledge, was not fully researched before (instead, other types of shift invariant wavelet transforms, such as those cited in manuscript [1] sec. 1 "Introduction", were developed). The shift variance theorem, formalized in the sec. 2 "Method", was implemented and can be used both for efficiency reasons (e.g. increasing the absorption rate of a streaming data set analysis) and to prediction purposes (dramatically reducing the number of wavelet coefficients to be estimated in the attempt to forecast the one step ahead DWT crystal).

The software has now been released for the first time. Within the authors' research group, its use has been initially intended to research purposes. A projected pathway, useful to achieve industrial impact using the software, is outlined in manuscript [1] sec. 1 "Introduction". As such, an efficient SaaS application, suitable to provide DWT calculation services, is presently under development.

The implementation of the Daubechies FWT and its SVT variant are contained in a single header file, in order to allow any developer to use it as a core component of applications aimed to perform analysis and prediction of streaming time series. Considering the increasing number of devices connected to the internet, such as IoT sensors and surveillance devices, capable of generating streaming time series, we believe that an efficient method of performing wavelet analysis (such as the one here proposed), could lead to the creation of commercial IaaS of PaaS applications, similar to the one we are currently developing.

5. Conclusions

After having explored and formalized the shift variance properties of the Fast Wavelet Transform performed on streaming univariate datasets, we developed a C++ implementation of the FWT and its SVT variant, and integrate it in a prediction system named Wa.R.P., in order to test the forecasting accuracy of a discrete wavelet transform assisted machine learning framework.

Satisfactory results, obtained in the case study named "Bitcoin-USD hourly exchange rates prediction", suggest that the reduced

Please cite this article in press as: Stocchi M., Marchesi M. Fast wavelet transform assisted predictors of streaming time series. SoftwareX (2017), https://doi.org/10.1016/j.softx.2017.09.006.

Table A.1
List of files and their content (“Notes”). “Artificial Neural Networks” namespace gathers a template based implementation of generic multilayer perceptrons; “Financial Data” namespace gathers classes and functions for basic manipulation of financial data; “Fast Wavelet Transform” provides the classes needed to perform the FWT and its reduced variant; “Inference engine” files contain the classes and functions developed for prediction implementation and case-study purposes; “Test” files are “.cpp" files to be executed alternatively, in order to perform single different type of tests and to reproduce the present research.

Artificial neural networks	Notes
DSPX_ann_def.h	Artificial NN and typedefs
DSPX_ann_helper.h	Random, math, statistic, functors
DSPX_ann_neuron.h	Generic artificial neuron template class
DSPX_ann_layer.h	Generic layer template class
DSPX_ann_layer_input.h	Artificial Neural Networks input layer
DSPX_ann_neuron_perceptron.h	Rosenblatt Rumelhart Perceptron
DSPX_ann_neuron_output.h	Output neuron template class
DSPX_ann_layer_perceptron.h	Layer spec. hosting Perceptrons
DSPX_ann_layer_output.h	Layer spec. hosting Output Neurons
DSPX_ann_network.h	Neural Nets variadic template class
DSPX_ann_network_perceptron.h	Generic Multilayer Perceptron
DSPX_ann_network_help.h	Helper functions, test and train MLPs
Financial data	**Notes**
DSPX_financial_convert.h	Time conversions for financial strips
DSPX_financial_bar.h	Financial bar tuple and I/O
DSPX_financial_data.h	Financial data handler and I/O
Fast wavelet transform	**Notes**
DSPX_fast_wavelet_transform.h	FWT, Shift Variance Theorem
Inference engine	**Notes**
DSPX_help.h	Standard streams and memory helpers
DSPX_engine.h	Predictors and WARP inference engine
Test	**Notes**
T1.cpp	Test FWT and Inverse FWT
T2.cpp	Test Efficiency FWT, reduced FWT
DSPX_predictor.cpp	WARP system main
DSPX_benchmark_ann.cpp	ANN benchmark system
DSPX_benchmark_svm.cpp	SVM benchmark system
DSPX_benchmark_wdnn.cpp	WDNN benchmark system
Singletons	**Notes**
DSPX_singletons.cpp	Singletons — include in build everytime
Precompiled header	**Notes**
stdafx.h	Precompiled header

FWT, despite being a decimated and shift variant operation, can be used to preprocess digital shifting signals to be input to statistical estimators or regressor machines.

Finally, the implementation of the shift variance theorem allows to perform FWT operations of 1D streaming time series in a very efficient way, greatly reducing the number of operations otherwise carried out using the classical FWT algorithm.

Appendix. List of files and instructions to compile

In Table A.1 we list all the files containing the published code. Such software, as well as all the supporting files listed next, are hosted on a Github repository and available for download at the following URL:

https://github.com/marcoStocchi/DSPX_Fast_wavelet_transform_assisted_predictors_of_streaming_time_series.

The `.cpp` test files (`T1.cpp`, `T2.cpp`, `DSPX_predictor.cpp`) are main routines testing several facets of the DWT and its SVT implementation, as well as to reproduce the case study: only one of them should be included in the build when compiling the solution.

The `DPSX_singletons.cpp` instantiates singleton objects and global variables, and must be always included in the build.

References

[1] Stocchi M, Marchesi M. Fast wavelet transform assisted predictors of streaming time series. Digit Signal Process SoftwareX 2017. http://dx.doi.org/10.1016/j.dsp.2017.09.014.

[2] King DE. Dlib-ml: A machine learning toolkit. J Mach Learn Res 2009;10:1755–8.

Please cite this article in press as: Stocchi M., Marchesi M. Fast wavelet transform assisted predictors of streaming time series. SoftwareX (2017), https://doi.org/10.1016/j.softx.2017.09.006.