Review hess-2019-565

TITLE

Two-stage Variational Mode Decomposition and Support Vector Regression for Streamflow Forecasting

RECOMMENDATION

Major revision

ASSESSMENT

This paper introduces a variational mode decomposition (VMD)-based support vector regression (SVR), i.e., VMD-SVR, model for multi-step ahead streamflow forecasting. The authors strive to address the 'boundary effects' problem common to many time series decomposition approaches that are typically coupled with data-driven models such as VMD, ensemble empirical mode decomposition (EEMD), singular spectrum analysis (SSA), wavelet transforms (WT), etc. outlined in earlier studies (Du et al., 2017; Maheswaran and Khosa, 2012; Quilty and Adamowski, 2018; Wang and Wu, 2016; Zhang et al., 2015). This is a worthwhile problem to address due to the growing interest in coupling these decomposition methods (VMD, EMD, SSA, WT, etc.) with data-driven models for hydrological forecasting and the vast majority of studies that overlook the impact of boundary effects on hydrological forecasting performance. Many of the just mentioned studies point out flaws in existing strategies for coupling decomposition methods with data-driven models and some go on to identify potential solutions.

In in this paper, the authors put forth their own approach for addressing boundary effects. The authors claim that the main benefits of their proposed approach include that it *"…can reduce the boundary effects, save the modelling time, and improve the prediction performance. This practical streamflow forecasting framework can be outlined as follows:*

(1) Divide the entire streamflow data into training and validation sets and decompose each of these two sets separately into signal components. This procedure avoids using the validation information for training purposes.

(2) Combine the predictors of individual signal components into a final predictors, and select the original streamflow data as the prediction target in order to build only one optimized prediction model.

(3) Generate training and validation samples and divide the validation samples into development and testing samples. Mix and shuffle the training and the development samples to optimize the prediction model, and reduce the boundary effects."

Throughout the MAJOR COMMENTS section below, I raise several issues with how the authors' proposed approach actually satisfies these points. In my opinion, I think there is much clarification required on the authors' part to demonstrate that they adequately fulfill these points (in a way that is meaningful for operational forecasting problems, which the present study appears to be concerned with). In particular, the authors' methodology for how they decompose the time series using VMD (and other comparative approaches) and use it in training and validating their proposed VMD-SVR (and comparative) method(s) is not entirely clear. Out of all issues raised in this review, this point needs the most attention.

Nonetheless, I find that the paper is well-written, is properly structured, and is supported by appropriate figures. The references are sufficient. The analysis carried out in the paper is reasonable but the validity of the paper's results, in terms of how useful the results are to those concerned with developing

operational forecasting models, largely depends on how the authors carried out decomposition of the time series (using VMD, DWT, etc.) and used it to develop the forecasting models.

If the authors can adequately address each of the comments/suggestions mentioned below, I would be happy to re-evaluate my current stance on the paper's suitability for being published in Hydrology and Earth System Sciences. In my opinion, the paper should not be published in its current form.

MAJOR ITEMS

In the Introduction, the authors mention how their *"…proposed scheme can reduce the boundary effects, save the modelling time, and improve the prediction performance. This practical streamflow forecasting framework can be outlined as follows:*

(1) Divide the entire streamflow data into training and validation sets and decompose each of these two sets separately into signal components. This procedure avoids using the validation information for training purposes.

(2) Combine the predictors of individual signal components into a final predictors, and select the original streamflow data as the prediction target in order to build only one optimized prediction model.

(3) Generate training and validation samples and divide the validation samples into development and testing samples. Mix and shuffle the training and the development samples to optimize the prediction model, and reduce the boundary effects."

Some comments on each of these points are given below:

Point 1

To decompose the validation data, the authors imply that they append one validation record at a time onto the calibration dataset (and any previous validation data) then perform VMD. The authors do this for each validation record, keeping the VMD components for each previous validation record static (this I my interpretation, the latter assertion was not specifically mentioned by the authors, at least that I could find). This appears to avoid boundary effects in the validation data due to the 'future data' issue; however, there are two major issues with this approach:

- The first issue is that VMD is sensitive to the addition of new data. I.e., by adding an additional data point to a time series and performing VMD creates inconsistencies between the intrinsic mode functions (IMFs) with the appended data and the IMFs prior to the appended data. Sometimes, these inconsistencies can be very large and tend to be largest at the edges of a time series, the most important time series observations in real-world forecasting applications (see example R script attached at the end of this review for the sensitivity of VMD to the additional of new data points).
- Because of this last point, the parameters of a model calibrated on initial IMFs generated by VMD that are then fed with updated IMFs based on the newly appended data may need to be updated to account for these newly introduced errors not seen during model calibration. This begs the question of whether each time the IMFs are updated whether the model should be updated too. Which goes against the authors' desire to implement a computationally-efficient forecasting method. Perhaps the authors may wish to consider a Kalman Filter to update their model

parameters if they follow such an approach (as opposed to completely re-training the model). The Kalman Filter could be used to update the model parameters at each time step or at larger intervals.

- The second issue is that VMD is shift-variant, meaning that performing VMD on lagged versions of the same time series leads to distortions in the IMFs derived by the VMD at the same times. This further exacerbates the issue raised above in terms of calibrating and validating a data-driven model based on using time-lagged inputs that are decomposed via VMD (see example R script attached at the end of this review that demonstrates the shift-variance problem in VMD).

Point 2

Although there are numerous studies that have considered forecasting each IMF (in EEMD, VMD, etc. based models) separately (and summing their constituent forecasts to obtain the final forecast), the authors should note that in the literature other studies have also built forecasts using, for instance, all wavelet-decomposed time series in a single forecasting model (Maheswaran and Khosa, 2013; Quilty and Adamowski, 2018). Many other examples of this approach can be found in the literature. It is suggested that the authors 'downplay' this feature of their framework as being something new or different.

Point 3

In terms of the mix and shuffle approach used to the training and validation data in the VMD-SVR models:

It is very difficult (for me) to see how taking all but the last 120 records of the red line in Figure 8 (b) (i.e., the development set) and randomly shuffling it with the red line from Figure 8 (a), (the training set) would lead to such a high performance on the last 120 records in Figure 8 (b) (i.e., the test set) as noted in Figure 9 and 11. Especially, when it appears that the training and combined development and test sets have completely different distributions (with the training set having a larger number of records).

It seems like something is missing here. To have a mean NSE of 0.2 for standard SVR and a mean NSE of nearly 1 for SSA-SVR, VMD-SVR, and DWT-SVR (Figure 11) is a sign that something is potentially awry with the decomposition process and its division into training, development, and test sets. A modest increase in NSE would make sense (between SVR and VMD-SVR) but such a large discrepancy between the standard SVR and the coupled SVR approaches (SSA-SVR, VMD-SVR, and DWT-SVR) makes me think that important details of the decomposition of the time series and its partitioning into the different data sets is missing.

I think it is necessary for the authors (at the very least) to provide pseudo-code for how they decomposed the time series using EEMD, SSA, VMD, and DWT as well as how it was partitioned into training, development, and testing sets including how the mixed sampling approach was carried out. It would be ideal if the authors could provide the code that they used for these steps and (if possible) the time series used to develop the models. This would allow for the substantial difference in results (between SVR and VMD-SVR) to be validated.

Some other points include:

1. The authors seem to be familiar with the framework proposed by Quilty and Adamowski (2018) and note that these authors avoided boundary effects through their approach (Wavelet Data-Driven Forecasting Framework, WDDFF). Why did the authors not compare the WDDFF against their VMD approach? As it stands, each of the comparison methods (EEMD, SSA, and DWT) used

in this study are all impacted by the boundary effect. At the end of this review, the reviewer has included a MATLAB code for how the authors could obtain boundary-corrected wavelet and scaling coefficients through the maximal overlap discrete wavelet transform. If the authors use this script to decompose their input time series and include it in their SVR model, they can easily replicate the WDDFF from Quilty and Adamowski (2018).

2. Also, since the authors are following only a univariate time series forecasting problem without considering exogenous variables such as rainfall, evaporation, it would also seem plausible that they should compare their framework to simple time series methods such as ARIMA or even more appropriately, fractionally-differenced ARIMA (also known as Hurst-Kolmogorov processes, HKp) that are known to be suitable for forecasting time series with multiscale behaviour. The HKp method has been shown to be a useful method for monthly streamflow forecasting, with the potential to outperform common machine learning methods (k nearest neighbours, neural networks) (Koutsoyiannis et al., 2010).

3. Line 266-7: 'Therefore, only a few decomposition values of the training set are affected by the boundary effects.' Can the authors validate this claim (i.e., via a formula or through an experiment)? How can one determine which training records in the various VMD components include boundary effects?

4. VMD has many tuning parameters that, by the discussion in section 2.1, seems to greatly impact VMD performance. How then is VMD more user-friendly than the MODWT, which only requires the selection of a decomposition level and wavelet filter (although not trivial), for which there are only a finite number? From what I can tell, the parameters in VMD (aside from the selection of the number of IMFs) can take on an infinite number of values…

5. I think Line 320 should be re-cast in light of the fact that selecting the right combination of VMD parameters is technically more computationally-intensive than for the DWT or MODWT.

6. For Experiment 5, only odd numbered lead times were considered (3, 5, 7, and 9 months ahead). Why were even numbered lead times (2, 4, etc. months ahead) not considered?

7. Section 3.4: which open-source software was used for EEMD, SSA, VMD, and DWT?

8. Given that a Bayesian approach (BOGP) was used for SVR hyper-parameter optimization, could it not also be used to select the VMD-related parameters? One would think that you could use the BOGP to optimize both VMD and SVR parameters at once. If possible, I think it would be

interesting for the authors to consider this. If it is not feasible, a short discussion on why it is not feasible would be interesting.

9. Why was six-fold cross-validation selected for hyper-parameter optimization (why not 3, 5, or 10-fold cross-validation)?

10. Normally one has to set a range for the different hyper-parameters in the BOGP approach. What range was set for the various SVR hyper-parameters? It would be good to include what guided your selection of these particular ranges.

11. Line 495: Figure 8 (a) – I find it hard to agree with the statement that the training data is 'barely affected by the boundaries'. Between record 550 and 555 there is a difference between the red and blue lines of ~ 2 $*10^8$ m3! I think one can hardly dismiss this as being a small difference…I suggest acknowledging this rather large discrepancy as something significant.

    In fact, this is one of the issues of VMD, EEMD, etc. They are not shift-invariant and are sensitive to the addition of new observations (see supporting R code at the end of this review). I suggest the authors discuss in detail these disadvantage of VMD, especially in relation to the MODWT, which does not suffer from these problems and which may also be used, in a mathematically sound manner, to decompose multiscale and/or non-stationary time series into sub-time series capturing their prominent features (which potentially includes trends, periodicities, transients, etc.). I think more effort needs to be devoted to clearly identifying the particular advantages of using the VMD-based approach in this study over the MODWT (which again, does not suffer from such issues).

12. Figure 8(b) drives the above-mentioned point home much further… Comparing Figure 8 (a) and Figure 8 (b) it also appears to be the case that the validation data and training data come from distributions, too. It would seem logical that the forecasting model should be updated to account for this change through time (e.g., perhaps through a Kalman Filter)?

13. Figure 10: it would make it much easier to read if the authors reduced the marker size for the different methods in the scatter plots. For the hydrograph plots, it would be good to zoom in on a particular section, perhaps concentrating on the largest peak event?

14. Figure 18: Why was the standard SVR not included in this analysis? I think it should be included to show how much better the other approaches (DWT-SVR, VMD, SVR, etc.) are at longer lead times.

15. Line 755-756: ‘However, as far as we know, approaches of building a forecasting framework that is adapted to the boundary effect never be tried.’

    Are you sure? Quilty and Adamowski (2018) explored the boundary effect existing in popular wavelet-based decomposition methods (DWT, MODWT, etc.), then introduced a set of best practices that addresses these boundary conditions (completely) and implemented these best practices in a new forecasting framework tailored for real-world forecasting. I think one could say that their framework ‘adapted to the boundary effect’. In an earlier study by Maheswaran and Khosa (2012), the authors also discussed how to overcome some of the issues of the DWT by choosing a more appropriate wavelet decomposition method (à trous algorithm) that did not suffer from the same boundary conditions. I would also qualify their approach as ‘adapting to the boundary effect’. In my opinion, the texted quoted from Line 755-6 is not entirely true and should be revised.

MINOR ITEMS

- There are numerous grammatical and spelling errors. I did not note all of these issues. I recommend that the authors carefully check the paper for grammatical and spelling issues (e.g., Line 673 ‘…increase and decrease patterns…’ should read ‘…increasing and decreasing patterns…’).
- Line 709: ‘Orthometric’? I suggest trying to get your point across using different terms.
- Line 761: simulating or forecasting? This applies to the whole paragraph. Simulation and forecasting are generally regarded as different procedures in hydrology.
- Line 764: It is not clear what is meant by ‘…predictor-runoff relationship and the decomposition-runoff relationship’.
- Line 765: I think you mean accuracy instead of reliability (the latter is generally measured using probabilistic performance metrics). The same comment also applies to the sentence two lines below.
- Line 773: ‘lead’ not ‘leading’.
- Line 782: I would rephrase point ‘c’. Perhaps mention something along lines ‘Although some overfitting of the VMD-SVR occurs, the model still provides accurate out-of-sample forecasts’.
- Line 789-90: Such as…? It would be good to provide some ideas concerning how you think this can be realized.

REFERENCES

Du, K., Zhao, Y., Lei, J., 2017. The incorrect usage of singular spectral analysis and discrete wavelet transform in hybrid models to predict hydrological time series. J. Hydrol. 552, 44–51. doi:10.1016/j.jhydrol.2017.06.019

Koutsoyiannis, D., Yao, H., Georgakakos, A., 2010. Medium-range flow prediction for the Nile: a comparison of stochastic and deterministic methods. Hydrol. Sci. J. 53, 142–164. doi:10.1623/hysj.53.1.142

Maheswaran, R., Khosa, R., 2013. Wavelets-based non-linear model for real-time daily flow forecasting

in Krishna River. J. Hydroinformatics 15, 1022. doi:10.2166/hydro.2013.135

Maheswaran, R., Khosa, R., 2012. Comparative study of different wavelets for hydrologic forecasting. Comput. Geosci. 46, 284–295. doi:10.1016/j.cageo.2011.12.015

Quilty, J., Adamowski, J., 2018. Addressing the incorrect usage of wavelet-based hydrological and water resources forecasting models for real-world applications with best practices and a new forecasting framework. J. Hydrol. 563, 336–353. doi:https://doi.org/10.1016/j.jhydrol.2018.05.003

Wang, Y., Wu, L., 2016. On practical challenges of decomposition-based hybrid forecasting algorithms for wind speed and solar irradiation. Energy 112, 208–220. doi:10.1016/j.energy.2016.06.075

Zhang, X., Peng, Y., Zhang, C., Wang, B., 2015. Are hybrid models integrated with data preprocessing techniques suitable for monthly streamflow forecasting? Some experiment evidences. J. Hydrol. 530, 137–152. doi:10.1016/j.jhydrol.2015.09.047

EXAMPLE SCRIPTS

R script

```
#' Example R script to show the shift-invariancy and sensitivity to adding additional data points

#' when using VMD for to proivde inputs for operational forecasting tasks.



#' Note: run this script line by line instead of executing it in its entirety (as only the last plot

#' will be displayed in the latter case).


set.seed(123)

install.packages("EMD")

install.packages("vmd")

library(EMD) # this is so that we can use the sunspot series as a test case

library(vmd)

data("sunspot")

x <- sunspot$sunspot
```

```
################################################################################
##########

#' CHECK 1: Is VMD shift-invariant?

#' If yes, any shifted copy of an IMF from a VMD decomposition, similar to a shifted copy of the
#' original time series, should be maintained.
#'
#' For example, given the sunspot time series x (of length 386) we can generate a 1-step
#' advanced copy of the original time series as follows:
#'
#' x0 = x[1:385]
#' x1 = x[2:386] # this is a 1-step advanced version of x0
#'
#' Obviously, shift-invariancy is preserved between x0 and x1 since x0[2:385] = x1[1:384]
#'
#'
#' For shift-invariancy to be preserved for VMD, we should observe, for example, that the
#' VMD IMF1 components for x0 (x0.IMF1) and x1 (x1.IMF1) should be exact copies of one another,
#' advanced by a single step.
#'
#' I.e., x0.IMF1[2:385] should equal x1.IMF1[1:384] if shift-invariancy is
#' preserved.  However, this is not the case for VMD as shown below.  Interestingly, we see
#' a high level of error at the boundaries of the time series, of high importance in
#' operational forecasting tasks.
```

```
################################################################################################
##########

# lag/advance sunspot data by 1-step

x0 = x[1:385]

x1 = x[2:386]

# perform VMD on both original and advanced time series

x0.vmd = vmd(x0)$getResult()$u # default settings

x1.vmd = vmd(x1)$getResult()$u # default settings

# plot IMF1 for x0.vmd and x1.vmd

plot(x0.vmd[2:385,1],type='l')

lines(x1.vmd[1:384,1],col='red')

# plot error (which should be a straight line of 0s if shift-invariancy was preserved)

err = x0.vmd[2:385] - x1.vmd[1:384]

plot(err)

# check the level of error (as measured by the mean square error) between the IMF components

mse = mean(err^2)

################################################################################################
##########

#' CHECK 2: The impact of appending data points to a time series then performing VMD, analogous

#' to the case in operational forecasting when new data becomes available and an updated forecast
```

```
#' is made using the newly arrived data.
#'
#' Ideally, for forecasting situations, when new data is appended to a time series an some pre-
#' processing is performed, it should not have an impact on previous measurements of the pre-
#' processed time series.
#'
#' For example, if IMF1_1:N represents the IMF1, which has N total measurements and was derived
#' by applying VMD to x_1:N then we would expect that when we perform VMD when x is appended with
#' another measurement, i.e., x_1:N+1, resulting in IMF1_1:N+1 that the first 1:N measurements in
#' IMF1_1:N+1 are equal to IMF1_1:N.  In other words, IMF_1:N+1[1:N] = IMF1_1:N[1:N].
#'
#' Again, we see that this is not the case. Appending an additional observation to the time series
#' results in the updated VMD components to be entirely different then the original (as of yet
#' updated) VMD components.  Interestingly, we see a high level of error at the boundaries of the
#' time series, of high importance in operational forecasting tasks.



################################################################################################
##########


# extend x with an additional measurement
x_1_385 = x[1:385]
x_1_386 = x[1:386]


# perform VMD on original and extended time series
x_1_385.vmd = vmd(x_1_385)$getResult()$u # default settings
x_1_386.vmd = vmd(x_1_386)$getResult()$u # default settings
```

```
# plot IMF1 for x_1_385.vmd and x_1_386.vmd
plot(1:386, x_1_386.vmd[1:386,1],type='l')
lines(1:385, x_1_385.vmd[1:385,1],col='red')

#' plot error (which should be a straight line of 0s if appending an additional observation has
#' no impact on VMD)
err.append = x_1_385.vmd[1:385] - x_1_386.vmd[1:385]
plot(err.append)

# check the level of error (as measured by the mean square error) between the IMF components
mse.append = mean(err.append^2)

# the problem gets exasperated if it is not a single time point that is updated, but several
x_1_300 = x[1:300]

# perform VMD on original and extended time series
x_1_300.vmd = vmd(x_1_300)$getResult()$u # default settings

# plot IMF1 for x_1_385.vmd and x_1_386.vmd
plot(1:386, x_1_386.vmd[1:386,1],type='l')
lines(1:300, x_1_300.vmd[1:300,1],col='red')

#' plot error (which should be a straight line of 0s if appending an additional observation has
#' no impact on VMD)
err.append.ext = x_1_300.vmd[1:300] - x_1_386.vmd[1:300]
plot(err.append.ext)

# check the level of error (as measured by the mean square error) between the IMF components
```

```
mse.append.ext = mean(err.append.ext^2)


# EOF
```

MATLAB script

```
% MATLAB code to calculate boundary-corrected wavelet coefficients using MODWT (should work in MATLAB R2016a and above)


X=rand(1000,1); % random time series

wname = 'db2'; % wavelet filter (change to suit needs)

[g, h] = wfilters(wname,'r'); % get reconstruction low ('g') and high ('h') pass filters

L = numel(g); % number of filter coefficients

J = 2; % decomposition level (change to suit needs)

L_J = (2^J - 1)*(L - 1) + 1; % number of boundary-coefficients at beginning of time series (remove these)

coefs = modwt(X, wname, J); % get second level MODWT wavelet and scaling coefficients

W_bc = coefs(1:J,L_J+1:end).'; % boundary-corrected wavelet coefficients

V_bc = coefs(J+1,L_J+1:end).'; % boundary-corrected scaling coefficients
```