UCR Time Series Classification Archive

Please reference as:

Welcome!

Dear Colleague,

If you are reading this, you are interested in using the UCR Time Series Classification Archive. This archive is a superset of, and completely replaces [8]. The current version, thereafter referred to as Fall 2018 expansion, will eventually replace Summer 2015 release [9]. The archive originally was born out of our frustration with papers reporting error rates on a single dataset, and claiming (or implicitly suggesting) that the results would generalize [6]. However, while we think the availability of previous versions of the UCR Archive has mitigated this problem to a great extent, it may have opened up other problems.

1) Several researchers have published papers on showing “we win some, we lose some” on the UCR Archive. However, there are many trivial ways to get “win some, lose some” type results on these datasets (For example, just smoothing the data, or generalizing from 1-NN to $k$-NN etc.). Using the archive can therefore apparently add credence to poor ideas (very sophisticated tests are required to show small but true improvement effects [3][7]). In addition Gustavo Batista has pointed out that “win some, lose some” is worthless unless you know in advance which ones you will win on! [4]. Dau et al. discuss this in great detail [10].

2) It could be argued that the goal of researchers should be to solve real world problems, and that improving accuracy on the UCR Archive is at best a poor proxy for such real world problems. Bing Hu has written a beautiful explanation as to why this is the case [2].

In spite of the above, the community generally finds the archive to be a very useful tool, and to date, more than 1,200 people have downloaded the UCR archive, and it has been referenced several hundred times.

We are therefore delighted to share this resource with you. We encourage you to read the paper accompanies this new archive expansion [10]. The password you need to unlock the data download is available in this document, read on to find it.

Best of luck with your research.

Eamonn, Anh and the Team
Data Format

Each of the datasets comes in two parts, a TRAIN partition and a TEST partition. For example, for the Fungi dataset we have two files, Fungi_TEST.txt and Fungi_TRAIN.txt. The two files will be in the same format, but are generally of different sizes. The files are in the standard ASCII format that can be read directly by most tools/languages. For example, to read the data of Fungi dataset into MATLAB, we can type...

```
>> TRAIN = load('Fungi_TRAIN.txt');
>> TEST = load('Fungi_TEST.txt');
```

...at the command line.

There is one time series exemplar per row. The first value in the row is the class label (an integer between 1 and the number of classes). The rest of the row are the data sample values. The order of time series exemplar carry no special meaning, and is in most cases random. A small number of datasets have class label starting from 0 or -1 by legacy.

![Fungi_TEST.txt](image)

This instance is in class 1

This instance is in class 2
Sanity Check

In order to make sure that you understand the data format, you should run this simple piece of code to test SyntheticControl dataset (you can cut and paste it, it is standard MATLAB).

Note that this is slow “teaching” code. To consider all the datasets in the archive, you will probably want to do something more sophisticated (indexing, lower bounding etc).

Nevertheless, we highly recommend you start here.
In this package we have produced a spreadsheet that gives basic information about the datasets (number of classes, size of train/test splits, length of time series etc).

In addition, we have computed the error rates for:

- Euclidean distance
- DTW, unconstrained
- DTW, after learning the best constraint in from the train set*
- Default rate (that is, the most probable class). To be consistent, we display default error rate, which is \((1 - \text{default_rate})\).

*Note that our simple method for learning the constraint is not necessary the best (as explained in the next slide).
Worked Example

We can use the Archive to answer the following question. Is DTW better than Euclidean distance for all/most/some/any problems?

As explained in [4], if DTW is only better on some datasets, this is not very useful unless we know ahead of time that it will be better. To test this we can build a Texas Sharpshooter plot (see [4] for details).

In brief, after computing the baseline (here, the Euclidean distance) we then compute the expected improvement we would get using DTW (at this stage, learning any parameters and settings), then compute the actual improvement obtained (using these now hardcoded parameters and settings).

When we create the Texas Sharpshooter plot, each dataset fall into one of four possibilities.

In our worked example, we will try to optimize the performance of DTW, looking only at the training data and predict its improvement (which could be negative), in a very simple way.

**Expected Improvement:** We will search over different warping window constraints, from 0% to 100%, in 1% increments, looking for the warping window size that gives the highest 1-NN training accuracy (if there are ties, we choose the smaller warping window size).

**Actual Improvement:** Using the warping window size we learned in the last phase, we test the holdout test data on the training set with 1-NN.

Note that there are better ways to do this (learn with increments smaller than 1%, use k-NN instead of 1-NN, do cross validation within the test set etc). However, as the next slides show, the results are pretty unambiguous even for this simple effort.
The results are strongly supportive of the claim “DTW better than Euclidean distance for most problems.”

We sometimes had difficulty in predicting when DTW would be better/worse, but many of the training sets are tiny, making such tests very difficult.

For example, 9 is BeetleFy, with just 20 train and 20 test instances. Here we expected to do a little better, but we did a little worse.

In contrast, for 66 (LargeKitchenAppliances) we had 375 train and 375 test instances, and where able to more accurately predict a large improvement.
(after plotting in MATLAB, the code is in Appendix A, you can zoom in to avoid the visual clutter seen to the right).
Suggested Best Practices/Hints

1. If you modify the data in anyway (add noise, add warping etc), please give the modified data back to the archive before you submit your paper (we will host it, and that way a diligent reviewer can test your claims while the paper is under review).

2. Where possible, we strongly advocate testing and publishing results on all datasets (to avoid cherry-picking), unless of course you are making an explicit claim for only a certain type of data (i.e. classifying short time series). In the event you don't have space in your paper, we suggest you create an extended tech report online and point to it. Please see [4] (esp. Fig 14) for some ideas on how to visualize the accuracy results on so many datasets.

3. If you have additional datasets, we ask that you donate them to the archive in our simple format.

4. When you write your paper, please make reproducibility your goal. In particular, explicitly state all parameters. A good guiding principle is to ask yourself Could a smart grad student get the exact same results as claimed in this paper with a days effort?. If the answer is no, we believe that something is wrong. Help the imaginary grad student by rewriting your paper.

5. Where possible, make your code available (as we have done), it will makes the reviewers task easier.

6. If you are advocating a new distance/similarity measure, we strongly recommend you test and report the 1-NN accuracy (as we have done). Note that this does not preclude the addition of other of tests (we strongly encourage additional test), however the 1-NN test has the advantage of having no parameters and allowing comparisons between methods.

7. Note that for 85 datasets of Summer 2015 release, the data are z-normalized by legacy. Paper [7] explains why this is very important. For 43 datasets of Fall 2018 expansion (this release), data are kept as is unless they were already z-normalized by donating source.
Suggested Reading


6. Rakthanmanon, Thanawin, et al. “Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping.” *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7.3 (2013): 10. If you are claiming that DTW is too slow… Maybe, but read this first.


Appendix A:
Sharpshooter Plots

Here is the code we used to produce the Sharpshooter plots.

```matlab
function[]=plot_texas_sharpshooter() % Compute a Texas Sharpshooter plot of DTW over Euclidean Distance. See
% Batista, Wang and Keogh (2011) A Complexity-Invariant Distance Measure
% for Time Series. SDM 2011
% Last updated October 2018 by Hoang Anh Dau

% For example, if we want to construct the figure for comparison between
% Euclidean distance (ED) and DTW
% expected_accuracy_gain = DTW_train_accuracy / ED_train_accuracy
% actual_accuracy_gain = DTW_test_accuracy / ED_test_accuracy
% Because we are using 1-NN classifier, there is no training; therefore
% for ED, we use test result only; for DTW, there is train result from
% leave-one-out cross-validation to learn the warping constraint.

% read in result sheet
result_file = 'texas_plot_2018.csv';
result = importdata(result_file, ',', 1);
error_rates = result.data;

% Note that the order of texas_names and texas_values must be the same.
texas_names = result.textdata(2:end, 1);
% Note that here we convert error to accuracy, by subtracting from 1
texas_values = 1 - error_rates;

expected_accuracy_gain = texas_values(:,2)./texas_values(:,1);
actual_accuracy_gain = texas_values(:,3)./texas_values(:,1);

% Produce plot just so we can get Xlim and Ylim
plot(expected_accuracy_gain,actual_accuracy_gain,'r.');
Xaxis = get(gca,'XLim');
Yaxis = get(gca,'YLim');
clf
hold on;
axis square;

% Bottom left quadrant
patch([Xaxis(1) 1 1 Xaxis(1)],[Yaxis(1) Yaxis(1) Yaxis(1)], [0.9843 0.8471 0.5765]);
% Top right quadrant
patch([1 1 Yaxis(2) Xaxis(2)],[1 1 Yaxis(2) Yaxis(2)], [0.9843 0.8471 0.5765]);
plot(expected_accuracy_gain,actual_accuracy_gain, 'r.');
xlabel('Expected Accuracy Gain');
ylabel('Actual Accuracy Gain');

% plot with symbol as number
for i = 1: length(texas_values(:,1))
    text(expected_accuracy_gain(i),actual_accuracy_gain(i),int2str(i))
end

% plot with symbol as dataset name
% for i = 1: length(texas_values(:,1))
% text(expected_accuracy_gain(i),actual_accuracy_gain(i),texas_names(i,:), 'rotation',+30)
% end
end
```
Here the result summary file for making the Texas Sharpshooter plot.

`texas_plot_2018.csv`

- First column is dataset name
- Second column is Euclidean distance test error rate
- Third column is DTW train error rate
- Last column is DTW test error rate
The Password

• As noted above. My one regret about creating the UCR Archive is that some researchers see improving accuracy on it as sufficient task to warrant a publication. I am not convinced that this should be the case (unless the improvements are very significant, or the technique is so novel/interesting it might be of independent interest).

• However, the archive is in a very contrived format. In many cases, taking a real world dataset, and putting it into this format, is a much harder problem than classification itself!

• Bing Hu explains this nicely in the introduction to her paper [2], I think it should be required reading for anyone working in this area.

• The password is the missing words from this sentence “Why would ******* use the archive and not acknowledge it?"

• The sentence is on the second page of [10]. The paper is available for download on the UCR Archive webpage or at https://arxiv.org/abs/1810.07758
I am somewhat bemused by the hundreds of papers that use the UCR Archive, but do not acknowledge or thank the archivists.

Many such papers thank funding agencies, people that donated CPU time, friends that gave feedback etc. But many of these papers could not have been written without access to dozens of labeled time series datasets.

These dozens of labeled datasets were provided, completely for free! And these datasets represent (now) at least a thousand hours of work by my students and collaborators, to create or collect, to clean and annotate, to compute benchmarks etc.

It does seem like an acknowledgment would be classy ;-)

Acknowledgments

The authors would like to thank Prof. Eamonn Keogh and all the people who have contributed to the UCR time series classification archive for their selfless work. We also thank the anonymous reviewers for their valuable advice.

This work has been supported by Major Project of High Resolution Earth Observation System of China (Grant No.03-Y20A04-9001-15/16), the CNES TOSCA-VEGIDAR Program, and CAS-CNRS Joint Doctoral Promotion Program.

References


About the baseline results reported – Before you ask

- Did you z-normalize the data before passing to the algorithm?
- There can be different implementations of DTW. Some implementations divide the distance by the warping path length; some use a different step patterns etc. We use MATLAB implementation of DTW [r1].
  
  ```matlab
  dist = dtw(time_series_1, time_series_2, window_size, ’squared’);
  ```
- We use MATLAB implementation of k-NN [r2]
  
  ```matlab
  mdl = fitcknn(train_data, train_label, ’Standardize’, 0, ’NSMethod’, ’exhaustive’);
  ```
- We use leave-one-out cross-validation to learn the warping constraint
  
  ```matlab
  cross_validation = crossval(mdl, ’LeaveOut’, ’on’);
  ```
- For constrained warping, if the percentage of time series length results in a real number, you can round up or round down. We round up.
- We round the error rate to four decimal places. For a more comprehensive result comparison and other resources, we recommend the UEA & UCR Time Series Classification Repository [r3].

[r1] https://www.mathworks.com/help/signal/ref/dtw.html
About the baseline results reported –
How we handle special cases

• For time series of different lengths:
  ➢ In storing data: We pad NaN (to the end) to the length of the longest time series. This makes it convenient when loading data into MATLAB.
  ➢ In computing baselines: We add low amplitude random numbers (to the end) to the length of the longest time series to make all time series of equal length.

  % pad_len is the length of the padding
  time_series = [time_series, rand(1, pad_len)/1000];

• For time series with missing values
  ➢ In storing data: Missing values are represented with NaN (if NaN is at the end of the time series, it is not real missing values).
  ➢ In computing baselines: We use linear interpolation.

  time_series = fillmissing(time_series, 'linear', 2, 'EndValues', 'nearest');
43 datasets added in Summer 2018

The figures follow are intended to offer a quick inspection of the data. For readability, depending on the scenario, the data may be normalized or may be not, the number of exemplars per class may be one, three or many.
ACSF1

One exemplar per class, with z-normalization
Three exemplars per class, without z-normalization
AllGestureWiimoteY

Three exemplars per class, without z-normalization.
Three exemplars per class, without z-normalization
Three exemplars per class, with z-normalization.
Chinatown

Three exemplars per class, with z-normalization
Crop

Three exemplars per class, with z-normalization
DodgerLoopDay

One exemplar per class, with $z$-normalization.
DodgerLoopGame

One exemplar per class, without z-normalization
One exemplar per class, without z-normalization
EOGHorizontalSignal

Three exemplars per class, with z-normalization

Class 1

Class 2

Class 3

Class 4

Class 5

Class 6

Class 7

Class 8

Class 9

Class 10

Class 11

Class 12
**EOGVerticalSignal**

Three exemplars per class, with z-normalization
EthanolLevel

Three exemplars per class, with z-normalization
FreezerRegularTrain

Three exemplars per class, with z-normalization
FreezerSmallTrain

One exemplar per class, with z-normalization
Fungi

One exemplar per class, with z-normalization
Three exemplars per class, without z-normalization
Three exemplars per class, without z-normalization
Three exemplars per class, without z-normalization
GesturePebbleZ1

Three exemplars per class, without z-normalization
GesturePebbleZ2

Three exemplars per class, without z-normalization
Three exemplars per class, with z-normalization

Class 1

Class 2

Left) GunPoint recording of 2003, right) GunPoint recording of 2018. Top) Ann Ratanamahatana, bottom) Eamonn Keogh. The female and male actors are the same individuals recorded fifteen years apart.
GunPointMaleVersusFemale

Three exemplars per class, with z-normalization
GunPointOldVersusYoung

Three exemplars per class, with z-normalization

Class 1

Class 2
One exemplars per class, with z-normalization
InsectEPGRegularTrain

Three exemplars per class, with z-normalization
InsectEPGSmallTrain

One exemplars per class, with z-normalization
Three exemplars per class, with z-normalization.
MixedShapesRegularTrain

Three exemplars per class, with z-normalization
MixedShapesSmallTrain

One exemplar per class, with z-normalization
PickupGestureWiimoteZ

Three exemplars per class, without z-normalization
PigAirwayPressure

One exemplar per class, with z-normalization
One exemplar per class, with z-normalization.
One exemplar per class, with z-normalization
PLAID

One exemplar per class, without z-normalization
PowerCons

One exemplar per class, with z-normalization

Class 1

Class 2
Rock

Three exemplars per class, with z-normalization
One exemplar per class, with z-normalization
One exemplar per class, with z-normalization
One exemplar per class, with z-normalization.
ShakeGestureWiimoteZ

Three exemplars per class, without z-normalization
SmoothSubspace

Thirty exemplars per class, with z-normalization
Three exemplars per class, with z-normalization.
85 datasets from Fall 2015 release

The figures follow are intended to offer a quick inspection of the data. For readability, depending on the scenario, the data may be normalized or may be not, the number of exemplars per class may be one, three or many.
Adiac

Three exemplars per class, with z-normalization
ArrowHead

Three exemplars per class, with z-normalization
Beef

Three exemplars per class, with z-normalization
BeetleFly

Three exemplars per class, with z-normalization.
BirdChicken

Three exemplars per class, with z-normalization.
Car

Three exemplars per class, with z-normalization
CBF

Three exemplars per class, with z-normalization
Chlorine Concentration

Three exemplars per class, with z-normalization
CinCECGTorso

Three exemplars per class, with z-normalization
Coffee

Three exemplars per class, with z-normalization.
Computers

One exemplar per class, with z-normalization
CricketX

One exemplar per class, with z-normalization
CricketY

One exemplar per class, with z-normalization.
One exemplar per class, with z-normalization
DiatomSizeReduction

One exemplar per class, with z-normalization
Three exemplars per class, with z-normalization.
Three exemplars per class, with z-normalization
Three exemplars per class, with z-normalization
Earthquakes

One exemplar per class, with z-normalization
ECG200

Three exemplars per class, with z-normalization
ECG5000

One exemplar per class, with z-normalization
ECGFiveDays

Three exemplars per class, with z-normalization
ElectricDevices

One exemplar per class, with z-normalization
FaceAll

One exemplar per class, with z-normalization
FaceFour

Three exemplars per class, with z-normalization
FacesUCR

One exemplar per class, with z-normalization
FiftyWords

One exemplar per class, with z-normalization
Fish

Three exemplars per class, with z-normalization
FordA

One exemplar per class, with z-normalization
One exemplar per class, with z-normalization.
GunPoint

Three exemplars per class, with z-normalization.
Ham

Three exemplars per class, with z-normalization
HandOutlines

Three exemplars per class, with z-normalization
Haptics

Three exemplars per class, with z-normalization
Herring

Three exemplars per class, with z-normalization.
InlineSkate

Three exemplars per class, with z-normalization
InsectWingbeatSound

One exemplar per class, with z-normalization
ItalyPowerDemand

Three exemplars per class, with z-normalization.
LargeKitchenAppliances

One exemplar per class, with z-normalization
Lightning2

One exemplar per class, with z-normalization
Lightning7

One exemplar per class, with z-normalization
One exemplar per class, with z-normalization
Meat

Twenty exemplars per class, with z-normalization
MedicalImages

One exemplar per class, with z-normalization.
MiddlePhalanxOutlineAgeGroup

Three exemplars per class, with z-normalization
Three exemplars per class, with z-normalization
MiddlePhalanxTW

Three exemplars per class, with z-normalization
MoteStrain

One exemplar per class, with z-normalization

Class 1

Class 2
One exemplar per class, with z-normalization
One exemplar per class, with z-normalization
OliveOil

Three exemplars per class, with z-normalization
OSULeaf

One exemplar per class, with z-normalization
Three exemplars per class, with z-normalization
Phoneme

One exemplar per class, with z-normalization
Plane

Three exemplars per class, with z-normalization
ProximalPhalanxOutlineAgeGroup

Three exemplars per class, with z-normalization

Class 1

Class 2

Class 3
Three exemplars per class, with z-normalization
ProximalPhalanxTW

Three exemplars per class, with z-normalization
RefrigerationsDevices

One exemplar per class, with z-normalization
ScreenType

One exemplar per class, with z-normalization
ShapeletSim

One exemplar per class, with z-normalization
ShapesAll

One exemplar per class,
with z-normalization
One exemplar per class, with z-normalization
Three exemplars per class, with z-normalization
Three exemplars per class, with z-normalization
StarLightCurves

Three exemplars per class, with z-normalization
Strawberry

Three exemplars per class, with z-normalization
One exemplar per class, with z-normalization
Three exemplars per class, with z-normalization
SyntheticControl

One exemplar per class, with z-normalization
One exemplar per class, with z-normalization
One exemplar per class, with z-normalization.
Trace

Three exemplars per class, with z-normalization
TwoLeadECG

Three exemplars per class, with z-normalization
TwoPatterns

One exemplar per class, with z-normalization
One exemplar per class, with z-normalization
One exemplar per class, with z-normalization
One exemplar per class, with z-normalization

Class 1

Class 2

Class 3

Class 4

Class 5

Class 6

Class 7

Class 8
UWaveGestureLibraryZ

One exemplar per class, with z-normalization.
Wafer

One exemplar per class, with z-normalization
Wine

Twenty exemplars per class, with z-normalization
WordSynonyms

One exemplar per class, with z-normalization
Worms

One exemplar per class, with z-normalization
WormsTwoClass

One exemplar per class, with z-normalization
Yoga

One exemplar per class, with z-normalization.