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 \textit{superset} of, and completely replaces [5]. Both [5], and this current Archive were born out of my frustration with papers reporting error rates on a single dataset, and claiming (or implicitly suggesting) that the results would generalize [6]. However, while I 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 “\textit{we win some, we lose some}” on the UCR Archive. However, there are many trivial ways to get “\textit{win some, lose some}” type results on these datasets (For example, just smoothing the data, or generalizing from 1NN to KNN etc.). Using the Archive can therefore \textit{apparently} add credence to poor ideas (very sophisticated tests are required to show \textit{small} but \textit{true} improvement effects [3]). In addition Gustavo Batista has pointed out that “\textit{win some, lose some}” is worthless unless you \textit{know in advance} which ones you will win on! [4].

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 are delighted to share this resource with you. The password you need available in this document, read on to find it.

Best of luck with your research.

Eamonn Keogh
Data Format

Each of the datasets comes in two parts, a TRAIN partition and a TEST partition. For example, for the synthetic control dataset we have two files, synthetic_control_TEST and synthetic_control_TRAIN. 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 two synthetic control dataset s into Matlab, we can type...

```matlab
>> TRAIN = load('synthetic_control_TRAIN');
>> TEST = load('synthetic_control_TEST');
```

...at the command line.

There is one data instance 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 values, and individual time series.

This instance is in class 1

This instance is in class 2
function UCR_time_series_test %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% (C) Eamonn Keogh %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TRAIN = load('synthetic_control_TRAIN'); % Only these two lines need to be changed to test a different dataset.
TEST = load('synthetic_control_TEST'); % Only these two lines need to be changed to test a different dataset.

TRAIN_class_labels = TRAIN(:,1); % Pull out the class labels.
TRAIN(:,1) = []; % Remove class labels from training set.
TEST_class_labels = TEST(:,1); % Pull out the class labels.
TEST(:,1) = []; % Remove class labels from testing set.
correct = 0; % Initialize the number we got correct
for i = 1 : length(TEST_class_labels) % Loop over every instance in the test set
    classify_this_object = TEST(i,:);
    this_objects_actual_class = TEST_class_labels(i);
    predicted_class = Classification_Algorithm(TRAIN,TRAIN_class_labels, classify_this_object);
    if predicted_class == this_objects_actual_class
        correct = correct + 1;
    end;
end;
% Report progress
disp([int2str(i), ' out of ', int2str(length(TEST_class_labels)), ' done'])
end;

%%%%%%%%%%%%%%%%% Create Report %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp(['The dataset you tested has ', int2str(length(unique(TRAIN_class_labels))), ' classes'])
disp(['The training set is of size ', int2str(size(TRAIN,1)),' and the test set is of size ', int2str(size(TEST,1)),'.'])
disp(['The time series are of length ', int2str(size(TRAIN,2))])
disp(['The error rate was ',num2str((length(TEST_class_labels)-correct )/length(TEST_class_labels))])
%%%%%%%%%%%%%%%%% End Report %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Here is a sample classification algorithm, it is the simple (yet very competitive) one-nearest neighbor using the Euclidean distance.
% If you are advocating a new distance measure you just need to change the line marked "Euclidean distance"
function predicted_class = Classification_Algorithm(TRAIN,TRAIN_class_labels, unknown_object)
best_so_far = inf;
for i = 1 : length(TRAIN_class_labels)
    compare_to_this_object = TRAIN(i,:);
    distance = sqrt(sum((compare_to_this_object - unknown_object).^2)); % Euclidean distance
    if distance < best_so_far
        predicted_class = TRAIN_class_labels(i);
        best_so_far = distance;
    end
end;
end;

>> UCR_time_series_test
1 out of 300 done
2 out of 300 done
...
299 out of 300 done
300 out of 300 done
The dataset you tested has 6 classes
The training set is of size 300, and the test set is of size 300.
The time series are of length 60
The error rate was 0.12
In this package we have produced an Excel file 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 the test set*

*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, 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 1NN 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 1NN.

Note that there are better ways to do this (learn with increments smaller than 1%, use KNN instead of 1NN, 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, 51 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 76 (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 any way (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 the data is z-normalized. Paper [7] explains why this is very important.
Suggested Reading


Appendix A: Sharpshooter Plots

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

So, the password is the three redacted words from this sentence “Every item that we ******** ## @@@@@@@@@ belongs to exactly one of our well-defined classes”, after you remove the two spaces.

The sentence is on the first page of [2].