Time Series Classification Utility
User Manual

Hüseyin Kaya

September 30, 2013

Contents

1 Introduction

Time Series Classification Utility (TSCU) is a simple MATLAB program that you can use it to classify time series by choosing a couple of alignment methods including Dynamic Time Warping (DTW), Constrained DTW (CDTW) and Signal Alignment via Genetic Algorithm (SAGA).

I decided to prepare TSCU during my PhD which is about creating a new time series alignment algorithm and its application to various real−world problems. There were (and there are still) a bunch of useful tools for time series alignment, but none of them seem to provide a general framework for classification [1]. I also wanted to create a useful website so that people searching for a time series classification task will find all crucial information quickly.

2 Installation

TSCU is freely available from GitHub. Majority of the code lies in just one MATLAB script: tscu.m. However it is recommended to fetch the whole repository (which is about 1Mb) in order to obtain a few dependent files. You can use the following command to download the repository:

```
git clone https://github.com/hkayabilisim/TSCU.git
```

This will checkout the repository into a new directory named TSCU. After downloading the source code, you should follow the following steps.

- In order to use the alignment methods DTW and constrained DTW, you should compile the mex file `dtw.c` by issuing the following command:

  ```
  mex dtw.c
  ```

- If everything goes well, you will have a new executable file with extension name begins with mex. In my MacBook Pro, its name is `dtw.mexmaci64`. If you have problems in compiling the mex file (you are very likely to face such problems, by the way), then you can look for precompiled binaries on TSCU repository.
- If you want to test the utility with the University of California, Riverside (UCR) time series repository, then you should request the dataset from Eamonn Keogh personally because I don't have permission to provide the dataset [1]. I strongly suggest you to have a copy of this large and diverse dataset if you want to do detailed analysis on alignment techniques.
- If you want to test the alignment algorithm Signal Alignment via Genetic Algorithm (SAGA), you should have Genetic Algorithm Toolbox of MATLAB. SAGA is optional, so if you don't have this toolbox, don't bother. You can still use DTW or constrained DTW for alignment.
- The cost function used in SAGA can be replaced with its MEX counterpart `'Jcost1.c'` which includes a single LAPACK call `dgesv`. You should first compile it by using the `'make.m'` script which includes a couple of alternative ways. On Mac OSX systems, Acceleration framework can be used as follows:

  ```
  mex  Jcost1.c  CLIBS="\$CLIBS -framework Accelerate" -largeArrayDims
  ```

  tscu_manual_verbatim06.out

  If, for some reason, you don't want to use Acceleration framework, then you can always compile LAPACK by yourself. In this case, compiling the MEX file is tricky. In addition to the lapack library, you have to build the BLAS and LAPACKE libraries. Finally, if you use `gfortran` compiler, then you should link the related library. The same is true for Intel Fortran compiler. Here are two example cases:

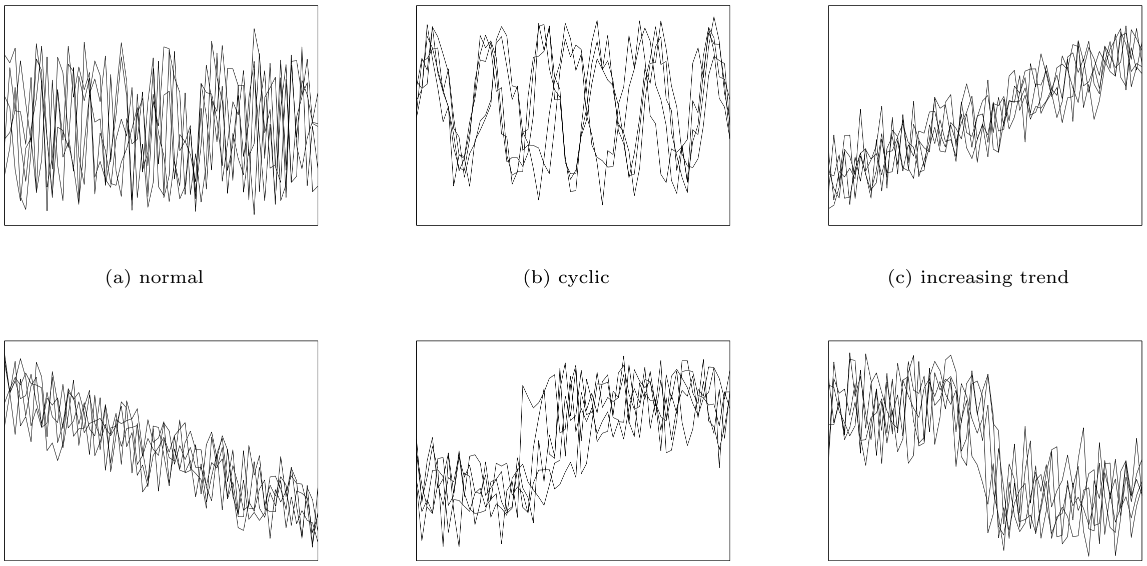
  ```
  mex Jcost1.c CLIBS="\$CLIBS -L lapack/MACOSXgfortran  -llapacke -llapack -lrefblas \
   -L     /opt/local/lib/gcc44 -lgfortran"
  mex Jcost1.c CLIBS="\$CLIBS -L lapack/Linuxifort -llapacke -llapack -lrefblas \
   -L /RS/progs/intel/lib/intel64 -lifcore -lirc"
  ```

  tscu_manual_verbatim05.out

  There is no significant performance gain if you choose to use the Acceleration framework. Yet another alternative would be to use ATLAS which I haven't tried yet.

3 Running

A straightforward way yo to test the installation is to run a few tasks. Let's start with the Synthetic Control dataset from University of California−Riverside (UCR) time series repository. This dataset is freely available on UCR time repository web page[1]. There are 6 different classes of time series each has length 60. There are totally 600 time series half of it is reversed for training. Some examples of the dataset are displayed in Figure 1.


(a) normal (b) cyclic (c) increasing trend

(d) decreasing trend (e) upward shift (f) downward shift

Figure 1: Some examples from 6 different control charts used in the synthetic control dataset.

After downloading training and testing set, you can classify the time series in the testing set by using the following command:

```
>> trn=load('synthetic_control_TRAIN');
>> tst=load('synthetic_control_TEST');
>> tscu(trn,tst)

Size of training set.....................: 300
Size of testing set......................: 300
Time series length.......................: 60
Classification method....................: 1-NN
Alignment method.........................: None
Overall Accuracy.........................: 0.880
Overall Error............................: 0.120
Producer Accuracy........................: 0.440   1.000   0.980   1.000   0.940   0.920
User Accuracy............................: 1.000   0.833   0.891   0.862   0.887   0.885
Kappa....................................: 0.856
Z-value..................................: 5.439
Confusion matrix
          1     2     3     4     5     6    UA    TO
    1    22     0     0     0     0     0 1.000   22
    2    10    50     0     0     0     0 0.833   60
    3     3     0    49     0     3     0 0.891   55
    4     4     0     0    50     0     4 0.862   58
    5     5     0     1     0    47     0 0.887   53
    6     6     0     0     0     0    46 0.885   52
   PA 0.440 1.000 0.980 1.000 0.940 0.920
```

[1]http://www.cs.ucr.edu/ eamonn/time_series_data/.

```
   TO    50    50    50    50    50    50          300
Time elapsed (sec).......................: 1.08
```

tscu_manual_verbatim01.out

Output of TSCU is pretty self explanatory. As you see from the output, TSCU does not use an alignment algorithm in its default form. Overall error in this case is 0.12 which is the same as the published error on UCR web site. TSCU also outputs confusion matrix which can sometimes be useful for further analysis.

If you want to use Dynamic Time Warping (DTW) as the alignment method for the same dataset, then you can append the following options:

```
>> trn=load('synthetic_control_TRAIN');
>> tst=load('synthetic_control_TEST');
>> tscu(trn,tst,'alignment','DTW')

Size of training set.....................: 300
Size of testing set......................: 300
Time series length.......................: 60
Classification method....................: 1-NN
Alignment method.........................: DTW
Overall Accuracy.........................: 0.993
Overall Error............................: 0.007
Producer Accuracy........................: 0.960   1.000   1.000   1.000   1.000   1.000
User Accuracy............................: 1.000   0.980   1.000   1.000   0.980   1.000
Kappa....................................: 0.992
Z-value..................................: 24.884
Confusion matrix
          1     2     3     4     5     6    UA    TO
    1    48     0     0     0     0     0 1.000    48
    2     1    50     0     0     0     0 0.980    51
    3     0     0    50     0     0     0 1.000    50
    4     0     0     0    50     0     0 1.000    50
    5     1     0     0     0    50     0 0.980    51
    6     0     0     0     0     0    50 1.000    50
   PA 0.960 1.000 1.000 1.000 1.000 1.000
   TO    50    50    50    50    50    50          300
Time elapsed (sec).......................: 5.77
```

tscu_manual_verbatim02.out

You can find more examples and their corresponding outputs on `html` directory.

4 Options

There are various options that you may want to use. Each option should be given with a key−value pair like `tscu(trn,tst,'Option1','value1','Option2','value2')`. Available options can be listed by running `help tscu` or `doc tscu` on MATLAB assuming that `tscu.m` is on the working directory or in the path.

4.1 `'Classifier'`

This option sets the classifier that is used to classify the instances in training and testing sets. Currently you can only set `'K−NN'` classifier. `'K−NN'` is also the default classifier with $K = 1$.

4.2 `'Alignment'`

This option specifies the alignment algorithm used in distance calculation between any two time series. The following values are available.

`'None'` *(default)* In this case no alignment takes place and usual Euclidean distance between two time series is taken as the distance.

`'DTW'` Standard Dynamic Time Warping is used in its original simple form without any lower bounding or bands. The implementation is based on the UCR Suite[2]. The code is written as a MATLAB MEX file to gain some speed. However one should compile the mex file *dtw.c* to be able to use it in TSCU.

`'CDTW'` Constrained Dynamic Time Warping in which the path is constrained in Sakoe−Chiba band. It is implemented again in the same mex file *dtw.c* however one should use the additional option `'DTWbandwidth'` to set the width of the band.

`'SAGA'` Signal Alignment via Genetic Algorithm. It uses smooth monotone functions suggested by Ramsay[2], but solves the best parameter set by using Genetic Algorithm[3]. It is more accurate but slower than others.

4.3 `'LogLevel'`

There are eight log levels whose range starts from "absolute silence" to "display everything": `'Emergency'`, `'Alert'`, `'Critical'`, `'Error'`, `'Warning'`, `'Notice'`, `'Info'` (*default*),and `'Debug'`.

4.4 `'SAGACostFunction'`

This options specifies the cost function used in SAGA. The speed of SAGA is directly affected with this choice. If you set this option but not choose SAGA as the alignment method, it will be silently ignored.

All cost functions calculates the distance below

$$d = ||x - y(w(t))||$$

where x and y are the time series and $w(t)$ is the warping path determined by the alignment algorithm. Warping function w is obtained by solving an ODE suggested by Ramsay [2]. The weight vector of the ODE is the free variable of the cost function. There are different implementations of the cost function.

`'Jcost0'` This implementation of the cost function first divides the unit interval and obtains a time vector whose length is equal to the length of time series. Then, it solves the ODE by using the weight vector in order to find the warping path. One of the time series is warped by using the warping function. Warping is achieved by using linear interpolation. Finally the Euclidean distance between the warped time series and the unwarped one is calculated. The related MATLAB implementation is shown below. Solution of ODE, interpolation and Euclidean distance are all conducted in MATLAB without depending any other software. So it is rather slow but portable i.e. you don't need to compile a MEX file, everything you need is in the package.

```
t=linspace(0,1,length(x));
J = @(s) norm(interp1(t,y,ramsay(t,s))-x);
```

tscu_manual_verbatim03.out

`'Jcost1'` This cost function is equivalent of `'Jcost0'` but it is rewritten as a MEX file (`'Jcost1.c'`) in order to gain some speed. The MEX file itself uses a LAPACK call (`dgesv`) to solve a linear system of equation. It is almost three times faster than `'Jcost0'`. However, you should be able to compile the MEX file `'Jcost1.c'`. Further information on compiling can be found in §Installation.

4.5 `'SAGAOptimizationMethod'`

The alignment method SAGA relies on minimization of the cost function defined in `'SAGACostFunction'`. By default, minimization of the cost function is achieved by using Genetic Algorithm. However some other optimization techniques may also be used.

[2]`http://www.cs.ucr.edu/~eamonn/UCRsuite.html`

`'GA'` *default* By default genetic algorithm is used to find the minimum point of the cost function.

`'Simplex'` By choosing this value, Nelder−Mead Simplex method is used as a minimization routing [4]. It requires an initial point which can be specified with `'InitialSolution'` option.

4.6 `'SAGABaseLength'`

It is the number of spline bases used in ODE proposed by Ramsay. Default value is 8.

4.7 `'SAGAInitialSolution'`

Some of the optimization algorithms specified in `'SAGAOptimizationMethod'` option may need an initial solution. For instance `'Simplex'` requires a starting point. By default it is set to zero vector with length `'SAGABaseLength'`.

4.8 `'CrossValidation'`

Not yet implemented.

4.9 `'MATLABPool'`

If your MATLAB distribution includes Parallel Computing Toolbox, then one can feed the name of parallel pool into TSCU so that any suitable loop is run parallel. If your computer has a multi−core CPU and you have this toolbox, setting this option to `'local'` will enable MATLAB to distribute the workload to the available cores. Currently, only the `'for'` loops related with K−NN classifier have been converted to `'parfor'` counterparts. If you choose K−NN and enable this option, it is high likely to get a speed−up of factor 4 by using a quad−core processor.

4.10 `'reportLineWidth'`

It defines the width of the first part of the report lines produced by the utility. The default value is 60.

4.11 `'trainingRatio'`

If the data is not already divided, it is divided into training and testing set.

4.12 `'DTWbandwidth'`

6 *default* This parameter is used when one choose `'CDTW'` as the alignment method. It is the width of the Sakoe−Chiba band defined in percentage. Setting it to 100 is the same effect as running DTW.

4.13 `'DisplayInputData'`

If it is set to `'yes'`, then both training and testing data is displayed.The default value is `'no'`.

4.14 `'DisplayAlignment'`

If you want to display the alignments between specific training and testing objects, then you can use this parameter. You should give a cell containing two row vectors corresponding to the indexes of training and testing samples. For instance, `'DisplayAlignment',{[1 3],[1]}` will display the alignments between the testing object 1 and the training objects 1 and 3. So we will have two alignments displayed.

5 Examples

In order to classify synthetic control dataset with default options you can use the following commands provided that you first downloaded the dataset:

```
trn=load('synthetic_control_TRAIN');
tst=load('synthetic_control_TEST');
tscu(trn,tst)
```

tscu_manual_verbatim04.out

You can specify the alignment algorithm by using the option `'alignment'`. For instance, to choose classic Dynamic Time Warping method, you may use `tscu(trn,tst,'alignment','DTW')`. In order to use constrained DTW, you can use: `tscu(trn,tst,'alignment','CDTW','DTWbandwidth',6)`.

6 Known issues

- TSCU can not use multi−channel time series. However, one can first reduce the number of channels to a single channel by using straightforward technique of summing the channels or by using some other dimensional reduction algorithms. Choosing the right approach really depends on the application, so I didn't implement such methods in TSCU. Channel reduction is left to the responsibility of the user.
- `'Jcost1'` is slower than `'Jcost0'` although the former is written in MEX.

References

[1] Keogh E., Zhu Q., Hu B., Hao Y., Xi X., Wei L., and Ratanamahatana C. A. The UCR time series classification/clustering homepage: `www.cs.ucr.edu/~eamonn/time_series_data`. Last checked on 3 Aug 2013.

[2] J. O. Ramsay and X. C. Li. Curve registration. *Journal of the Royal Statistical Society Series B-statistical Methodology*, 60(Part 2):351–363, 1998.

[3] Hüseyin Kaya and Şule Gündüz-Öğüdücü. SAGA: A novel signal alignment method based on genetic algorithm. *Information Sciences*, 228(0):113–130, 2013.

[4] JC Lagarias, JA Reeds, MH Wright, and PE Wright. Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM JOURNAL ON OPTIMIZATION*, 9(1):112–147, DEC 21 1998.