Efficient search of the best warping window for Dynamic Time Warping Supplementary material

Chang Wei Tan[1] Matthieu Herrmann[1] Germain Forestier[2,1] Geoffrey I. Webb[1] François Petitjean[1]

[1]Faculty of IT, Monash University, Melbourne, Australia – firstname.lastname@monash.edu

[2]MIPS, University of Haute Alsace, Mullhouse, France – firstname.lastname@uha.fr

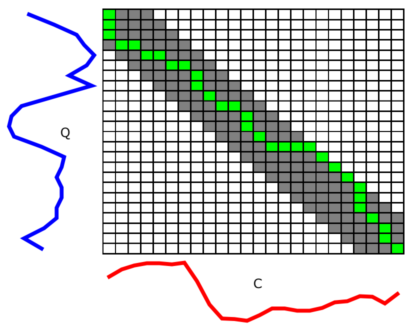

Figure A.1: Warping window of size 3.

A Warping window and cross-validation

An example of warping window and associated warping path is given in Figure A.1.

Algorithms A.1 and A.2 present the leave-one-out cross-validation (LOOCV) approach to learning the warping window.

Algorithm A.1: SOTAWWSEARCH($\mathcal{T}, LB$)

Input: Data $\mathcal{T}$, lower bound LB
Result: $w^\star$: the WW with lowest CV error

bestNErrors $\leftarrow |\mathcal{T}| + 1$
for $w \leftarrow 0$ **to** $L - 1$ **do**
  nErrors $\leftarrow 0$
  foreach $S \in \mathcal{T}$ **do**
    $nn \leftarrow$ LBNNSEARCH($S, \mathcal{T} \setminus \{S\}, w, LB$) **if** $nn.class \neq S.class$ **then**
      nErrors $\leftarrow$ nErrors+1
  if $nErrors < bestNErrors$ **then**
    $w^\star \leftarrow w$
    bestNErrors $\leftarrow$ nErrors
return $w^\star$

Algorithm A.2: LBNNSEARCH($S, \mathcal{T}, w, LB$)

Input: Query S, data $\mathcal{T}$, warping window w
Result: NN: Nearest neighbor of S in $\mathcal{T}$

NN $.dist \leftarrow +\infty$
foreach $T \in \mathcal{T}$ **do**
  if $\text{LB}_w(S, T) <$ NN $.dist$ **then**
    if $\text{DTW}_w(S, T) <$ NN $.dist$ **then** NN $\leftarrow T$
return NN

B Fail-Safe Experiment

Table 1 shows the warping window learnt by all the methods on some of the UCR benchmark datasets [1] using exhaustive search (searching all possible warping windows). Please refer to http://bit.ly/SDM18 for the full detailed results. Note that the results reported here are the actual warping window and not a percentage of the L (commonly done in the literature [1, 4, 2]. As expected, all the methods learnt the same warping window.

Figure B.1 shows the classification accuracy on the UCR Benchmark datasets [1] using the best warping window found for each individual method. Since all the methods are exact and that we are performing an exhaustive search (i.e. finding all possible warping windows $w = \{0, 1, 2, ..., L\}$), the best warping window found for each method is the same. Hence, the classification accuracy is the same. The only difference is the time which can be referred to Figure 6 in our main paper.

C Is it worth incorporating PrunedDTW within FastWWSearch?

It is interesting to examine if PRUNEDDTW could provide further improvements to our method. As the PRUNEDDTW algorithm [3] is able to speed up DTW computations, it is interesting to study if its incorporation into our algorithm could bring further benefits. Our method requires the different windows to be tested

	Best warping window learnt by the following methods				
Datasets	LB_Keogh	UCR Suite	LB_Keogh–PrunedDTW	UCR Suite–PrunedDTW	**FastWWSearch**
50words	24	24	24	24	24
Adiac	6	6	6	6	6
ArrowHead	0	0	0	0	0
Beef	0	0	0	0	0
BeetleFly	36	36	36	36	36
BirdChicken	33	33	33	33	33
CBF	14	14	14	14	14
Car	9	9	9	9	9
ChlorineConcentration	0	0	0	0	0
CinC_ECG_torso	10	10	10	10	10
Coffee	0	0	0	0	0
Computers	74	74	74	74	74
Cricket_X	31	31	31	31	31
Cricket_Y	47	47	47	47	47
Cricket_Z	15	15	15	15	15
DiatomSizeReduction	0	0	0	0	0
DistalPhalanxOutlineAgeGroup	1	1	1	1	1
DistalPhalanxOutlineCorrect	2	2	2	2	2
DistalPhalanxTW	0	0	0	0	0
ECG200	0	0	0	0	0
ECG5000	1	1	1	1	1
ECGFiveDays	0	0	0	0	0
Earthquakes	17	17	17	17	17
ElectricDevices	13	13	13	13	13
FISH	19	19	19	19	19
FaceAll	4	4	4	4	4
FaceFour	6	6	6	6	6
FacesUCR	16	16	16	16	16
FordA	2	2	2	2	2
FordB	6	6	6	6	6

Table 1: Learnt warping window from all the methods on some of the Benchmark datasets [1]

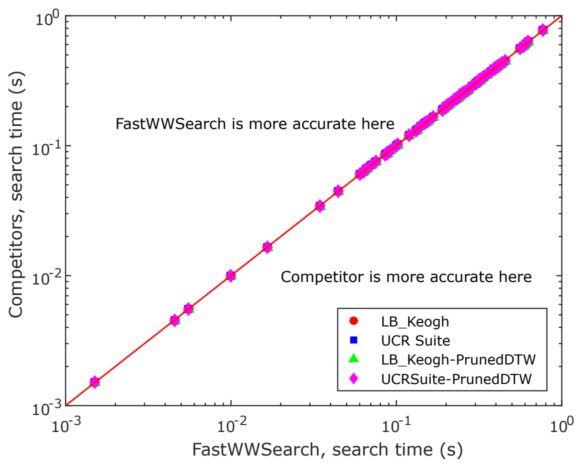

Figure B.1: Classification accuracy on the UCR Benchmark datasets [1] using the best warping window found for each method

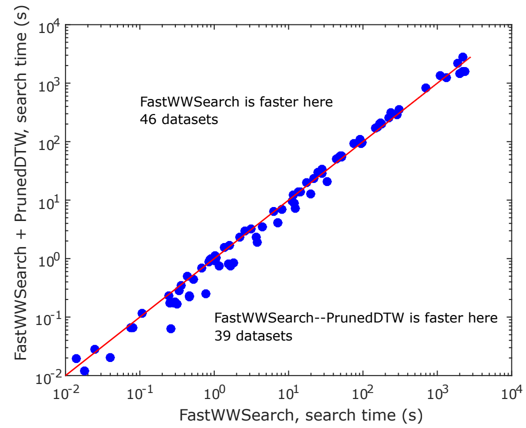

Figure C.1: Comparison of our method with the PRUNEDDTW implementation on the benchmark datasets

in descending order, in order to reuse previously calculated results as lower bounds to current ones. This is incompatible with a PRUNEDDTW search, which requires the windows to be assessed in ascending order, to use previous results as upper bounds to the next ones [3]. However, we could still use PRUNEDDTW whenever we have to calculate DTW, and use the Euclidean distance as a general upper bound [3]. This is how we incorporate PRUNEDDTW into FASTWWSEARCH.

Figure C.1 compares our original FASTWWSEARCH with a version incorporating PRUNEDDTW. The results show that both methods have similar running times, with the addition of PRUNEDDTW making it possible to gain some speed-up for low-runtime datasets (i.e. datasets that are either small or have short time series or both). Using FASTWWSEARCH vanilla seems to be even faster for high-runtime datasets. Overall, for approximately 55% of the UCR archive, FASTWWSEARCH vanilla is faster than having added PRUNEDDTW. This is because for high-complexity datasets, it seems that the added pruning power doesn't outweigh the additional computations of the upper bound that PRUNEDDTW requires. Overall, the take-home message here is that our method is totally compatible with PRUNEDDTW and that its incorporation is left at the discretion of the data practitioner, depending on their application.

References

[1] Y. CHEN, E. KEOGH, B. HU, N. BEGUM, A. BAGNALL, A. MUEEN, AND G. BATISTA, *The ucr time series classification archive*, 7 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

[2] V. NIENNATTRAKUL AND C. RATANAMAHATANA, *Learning dtw global constraint for time series classification*, arXiv preprint arXiv:0903.0041, (2009).

[3] D. SILVA AND G. BATISTA, *Speeding up all-pairwise dynamic time warping matrix calculation*, in Proceedings of the 2016 SIAM International Conference on Data Mining, SIAM, 2016, pp. 837–845.

[4] X. XI, E. KEOGH, C. SHELTON, L. WEI, AND C. RATANAMAHATANA, *Fast time series classification using numerosity reduction*, in Proceedings of the 23rd international conference on Machine learning, ACM, 2006, pp. 1033–1040.