Efficient Search of the Best Warping Window for Dynamic Time Warping

2018 SIAM International Conference on DATA MINING
3 May 2018

CW. Tan  M. Herrmann  G. Forestier  G.I. Webb  F. Petitjean
What is a Time Series?

- Collection of observations made sequentially, more intuitive **visually**
- Many data can be transformed into time series → **Satellite Image Time Series**

Array of numbers

0.348,0.245,0.142,0.183,0.203, 0.224,0.252,0.204,0.216,0.229, 0.241,0.177,0.211,0.254,0.360, 0.487,0.614,0.669,0.738,0.788, 0.815,0.807,0.817,0.817,0.821, 0.825,0.810,0.796,0.783,0.777, 0.685,0.667,0.591,0.566,0.467, 0.368,0.335,0.301,0.268,0.234, 0.238,0.233,0.262,0.261,0.247, 0.233

Every pixel represents a geographic area (Lat, Lon) on Earth
Dynamic Time Warping

• a.k.a. **DTW** – similarity function to align time series $O(L^2)$
• Nearest Neighbour Algorithm (**NN-DTW**) – Hard to beat [1]
• Used in many fields: Finance, Engineering, Speech Recognition, ...

Dynamic Time Warping

• Aligns two time series $Q$ and $C$ using Dynamic Programming
  • Build a cost matrix and solve:
    
    $$D_{Q,C}^{i,j} = \delta(q_i, c_j) + \min\left\{ \begin{align*} 
    D_{Q,C}^{i-1,j-1} \\
    D_{Q,C}^{i-1,j} \\
    D_{Q,C}^{i,j-1} 
    \end{align*} \right\}$$

  • where $\delta(q_i, c_j) = L_p -$ norm

    $$\text{DTW}(Q, C) = \left( D_{Q,C}^{m,n} \right)^{\frac{1}{p}}$$
Dynamic Time Warping

• Every possible alignment of \( Q \) and \( C \) is a warping path, \( \hat{p} \)
  \[ \hat{p} = [w_1, ..., w_K] \]
  \( w_k = (i, j) \) represents an association of \( q_i \leftrightarrow c_j \) aligned by DTW
• DTW(\( Q, C \)) finds the cheapest warping path ("best")
Warping Window

- Warping Window, $w$ is a global constraint on the alignment of DTW such that the elements of $Q$ and $C$ can only be mapped if they are less than $w$ apart, $w = \{0, \ldots, L\}$

\[
\text{Warping Window, } w
\]

- DTW with $w = L$
- DTW with $w = 3$
- DTW with $w = 0$

Full DTW
Warping windows, $w$
Euclidean Distance
Why learn the best warping window?

- **Strong** influence on accuracy
  - On CinC ECG torso dataset, error rate reduced from 35% to 7%
- **Outperforms** all existing time series classification (TSC) methods
  - State of the art – COTE and EE learn the best warping window for DTW
- **Speedup** DTW
  - Smaller $w$ means we don’t need to compute the full DTW matrix

---

How to learn the best warping window?

\[
\text{for } w = 0 \text{ to } L \text{ do } \quad \text{Parameter to NN-DTW algorithm}
\]

\[
\text{error} = 0
\]

\[
\text{for each } s \text{ in } T \text{ do } \quad \text{Leave One Out Cross Validation (LOO-CV)}
\]

\[
nn_s = \text{nn\_search}(s, T \backslash s, w) \quad \text{Can be any NN-DTW algorithm}
\]

\[
\text{if } \text{nn}_s.\text{class} \neq s.\text{class} \text{ then } \text{error}++
\]

\[
\text{if } \text{error} < \text{bestError} \text{ then } \quad \text{if}
\]

\[
\text{bestWW} = w
\]

\[
\text{bestError} = \text{error}
\]
Nearest Neighbour – DTW Search

• Naïve DTW Search

\[ \text{bestDist} = \infty \]

\[
\text{for each } c \text{ in } T \text{ do}
\]

\[
\text{dtwDist} = \text{DTW}(q, c, w)
\]

\[
\text{if } \text{dtwDist} < \text{bestDist} \text{ then}
\]

\[
\text{bestDist} = \text{dtwDist}
\]

\[
\text{nnIndex} = c.\text{index}
\]

• Lower Bound DTW Search

\[ \text{bestDist} = \infty \]

\[
\text{for each } c \text{ in } T \text{ do}
\]

\[
\text{lbDist} = \text{lowerBound}(q, c, w)
\]

\[
\text{if } \text{lbDist} < \text{bestDist} \text{ then}
\]

\[
\text{dtwDist} = \text{DTW}(q, c, w)
\]

\[
\text{if } \text{dtwDist} < \text{bestDist} \text{ then}
\]

\[
\text{bestDist} = \text{dtwDist}
\]

\[
\text{nnIndex} = c.\text{index}
\]


DTW Lower Bounds

• LB Kim

\[ \text{LB}_{\text{Kim}}(Q, C) = \max \left\{ \frac{|q_1 - c_1|}{|q_L - c_L|}, \frac{|q_{\text{max}} - c_{\text{max}}|}{|q_{\text{min}} - c_{\text{min}}|} \right\} \]

• LB Keogh

\[ \text{LB}_{\text{Keogh}}(Q, C) = \sum_{i=1}^{L} \begin{cases} (q_i - U_i)^2, & \text{if } q_i > U_i \\ (q_i - L_i)^2, & \text{if } q_i < L_i \\ 0, & \text{otherwise} \end{cases} \]


Reversing Query/Candidate in LB Keogh

- \( \max \left( \text{LB}_{\text{Keogh}}(Q, C), \text{LB}_{\text{Keogh}}(C, Q) \right) \)
- Increase tightness of LB Keogh
- Envelopes can be pre-computed
- We will show how we utilised all these “tricks” in our algorithm

Naïve approach learns the best warping window requires $\theta(N^2 L^3)$ operations.

Efficiently Search for the Best Warping Window of Any Time Series Dataset

Satellite Image Time Series

$N = 1,000,000$

$L = 46$
Related Methods

**UCR Suite**
- Improve efficiency of NN-DTW by minimising DTW computations
- 4 optimisation techniques
  - Early abandoning Z-Normalisation
  - Reordering early abandoning
  - Reversing query and candidate in LB Keogh
  - Cascading lower bounds
- Did not use to learn warping window but can be repurposed for this task

**Pruned DTW**
- Improve efficiency of DTW
- Compute an upper bound to minimise the computations by skipping the cells of the cost matrix that are larger
- Uses the DTW value with smaller $w$ as the upper bound to prune DTW with larger $w$
- Improvement for warping window search is minimal


Fast Warping Window Search for DTW

• a.k.a. FastWWS - An exact method
  • LazyAssessNN
  • FastFillINNTable

• Use links between different values of the loops

\[
\begin{align*}
\text{for } w = 0 \text{ to } L & \text{ do} \\
\text{error} &= 0 \\
\text{for each } s \text{ in } T & \text{ do} \\
\text{nn}_s &= \text{nn_search}(s, T\backslash s, w) \\
\text{if } \text{nn}_s.\text{class} \neq s.\text{class} & \text{ then } \text{error}++ \\
\text{if } \text{error} < \text{bestError} & \text{ then} \\
\text{bestWW} &= w \\
\text{bestError} &= \text{error}
\end{align*}
\]

These loops are independent

(1) For each warping window, \( w \)

(2) Find the nearest neighbour \( nn \) of each time series \( s \) in \( T\backslash s \)

All optimisation in the literature occurs here
Properties for FastWWS

1. Warping path can be valid for several windows
   • $w$ has a “validity”
   • skip computations of all valid $w$
   • Example:
     • Warping path is valid to $w = 6$
     • $\text{DTW}_{24}(Q, C) = \text{DTW}_{6}(Q, C)$
     • Skip all DTW from $w = [24, \ldots, 6]$

<table>
<thead>
<tr>
<th>$w$</th>
<th>...</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>...</th>
<th>23</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{DTW}_w(Q, C)$</td>
<td>...</td>
<td>8.82</td>
<td>8.36</td>
<td>8.04</td>
<td>8.04</td>
<td>...</td>
<td>8.04</td>
<td>8.04</td>
</tr>
</tbody>
</table>
Properties for FastWWS

1. Warping path can be valid for several windows

- Full DTW, $w = 24$
- $w = 6$
- $w = 5$
Properties for FastWWS

2. DTW is monotone with warping window
   - $\text{DTW}_w(Q, C) \leq \text{DTW}_{w-1}(Q, C)$

3. LB Keogh is monotone with warping window
   - $\text{LB}_{\text{Keogh}}_w(Q, C) \leq \text{LB}_{\text{Keogh}}_{w-1}(Q, C)$

New Lower Bounds to prune Nearest Neighbours before computing $\text{DTW}_w(Q, C)$

$$\text{DTW}_w(Q, C) \geq \text{DTW}_{w+1}(Q, C)$$
$$\text{LB}_{\text{Keogh}}_w(Q, C) \geq \text{LB}_{\text{Keogh}}_{w+1}(Q, C) \geq \text{LB}_\text{Kim}(Q, C)$$
FastWWS Intuition

- Efficiently fill up a NN table, giving the nearest neighbour of every time series for all windows
- Naïvely create the table using DTW, requires $\theta(N^2 L^3)$ operations

Prior approaches typically go from smallest to largest with a subset of windows

FastWWS goes from largest to smallest, fast enough to test all windows
FastWWS Intuition

- **FastWWS** goes from largest to smallest, applies to all or a subset of windows

- Large window validity for $\text{DTW}_L$ (Most of the time)
- No bounds are necessary
- DTW has not changed

Thus obtain $\text{DTW}_{w+k}$ and/or $\text{LB}_{\text{Keogh}}_{w+k}$ for “FREE” as the lower bound for $\text{DTW}_w$

**Tighter bounds for pruning**

$$\text{DTW}_w(Q, C) \geq \text{DTW}_{w+1}(Q, C)$$

$$\text{LB}_{\text{Keogh}}_w(Q, C) \geq \text{LB}_{\text{Keogh}}_{w+1}(Q, C) \geq \text{LB}_{\text{Kim}}(Q, C)$$

Only use the value at $w + k$ when available, no point in computing $\text{DTW}_{w+k}$ for $\text{DTW}_w$
FastWWS Intuition

- **FastWWS** goes from largest to smallest, applies to all or a subset of windows

1. If we find the nearest neighbour for a time series at window, $w = L$ and the warping path is valid to $w = 0$, then we only need to do 1 DTW computation.

2. When we calculate $DTW_w(Q, C)$, even if candidate $C$ is not the nearest neighbour of $Q$, we do not need to recompute $DTW_{w'}(Q, C)$ for all windows $w'$ that are valid.

![Graph showing DTW distance and warping paths for different windows.](image-url)
Lazy Nearest Neighbour Assessment

• Assess if a pair of time series \((Q, C)\), can be less than distance \(d\) for window \(w\)

• Postpones calculations for as long as possible
  1. First prune with lower bounds from larger window
  2. Try lower bounds of increasing complexity until
     a. \(A \text{LB}_w(Q, C) > d\)
     b. Calculated \(\text{DTW}_w(Q, C)\)

• When \(w\) decreases, any value previously calculated for a larger window becomes a lower bound for current \(w\), stored in a Cache, \(\mathcal{C}_{(Q,C)}\)
LazyAssessNN Algorithm

if cache\(_{Q,C}\) is empty do cache\(_{Q,C} = \text{LB}_\text{Kim}(Q, C)\)

if cache\(_{Q,C}\)\'.stoppedAt == DTW\(_{w+k}\) and \(_w\) is valid then
  if cache\(_{Q,C}\)\'.value ≥ \(_d\) return prunedByDTW else return cache\(_{Q,C}\)\'.value

if cache\(_{Q,C}\)\'.stoppedAt == LB_Kim or LB_Keogh\(_{w+k}\) then
  if cache\(_{Q,C}\)\'.value ≥ \(_d\) return prunedByLB

\(\text{cache}_{Q,C} = \text{LB}_\text{Keogh}_w(Q, C)\) if cache\(_{Q,C}\)\'.value ≥ \(_d\) return prunedByLB
\(\text{cache}_{Q,C} = \text{LB}_\text{Keogh}_w(C, Q)\) if cache\(_{Q,C}\)\'.value ≥ \(_d\) return prunedByLB
\(\text{cache}_{Q,C} = \text{DTW}_w(C, Q)\) if cache\(_{Q,C}\)\'.value ≥ \(_d\) return prunedByDTW

return cache\(_{Q,C}\)\'.value

1. First do LB Kim if hasn’t been done
LazyAssessNN Algorithm

2. Check lower bounds from previous window

if cache_{Q,C} is empty do cache_{Q,C} = LB_Kim(Q,C)
if cache_{Q,C}.stoppedAt == \text{DTW}_{w+k} and \ w \ is \ valid \ then
    if cache_{Q,C}.value \geq d \ return \ \text{prunedByDTW} \ else \ return \ cache_{Q,C}.value
if cache_{Q,C}.stoppedAt == \text{LB}_\text{Kim} \ or \ \text{LB}_\text{Keogh}_{w+k} \ then
    if cache_{Q,C}.value \geq d \ return \ \text{prunedByLB}

\text{cache}_{Q,C} = \text{LB}_\text{Keogh}_w(Q,C) \ \text{if} \ \text{cache}_{Q,C}.value \geq d \ \text{return} \ \text{prunedByLB}
\text{cache}_{Q,C} = \text{LB}_\text{Keogh}_w(C,Q) \ \text{if} \ \text{cache}_{Q,C}.value \geq d \ \text{return} \ \text{prunedByLB}
\text{cache}_{Q,C} = \text{DTW}_w(C,Q) \ \text{if} \ \text{cache}_{Q,C}.value \geq d \ \text{return} \ \text{prunedByDTW}
\text{return} \ cache_{Q,C}.value

DTW and \ LB \ Keogh \ from \ larger \ window \ (property \ 2 \ & \ 3)
LazyAssessNN Algorithm

3. Use DTW from previous window \((w + k)\) if current window \(w\) still valid (property 1)

4. If current window \(w\) is not valid

\[
\text{if } \text{cache}_{Q,C}\text{ is empty do} \quad \text{cache}_{Q,C} = \text{LB}_\text{Kim}(Q,C) \\
\text{if } \text{cache}_{Q,C}\text{.stoppedAt }== \text{DTW}_{w+k} \text{ and } w \text{ is valid then} \\
\quad \text{if } \text{cache}_{Q,C}\text{.value }\geq d \text{ return prunedByDTW else return cache}_{Q,C}\text{.value} \\
\text{if } \text{cache}_{Q,C}\text{.stoppedAt }== \text{LB}_\text{Kim} \text{ or } \text{LB}_\text{Keogh}_{w+k} \text{ then} \\
\quad \text{if } \text{cache}_{Q,C}\text{.value }\geq d \text{ return prunedByLB} \\
\text{cache}_{Q,C} = \text{LB}_\text{Keogh}_w(Q,C) \text{ if } \text{cache}_{Q,C}\text{.value }\geq d \text{ return prunedByLB} \\
\text{cache}_{Q,C} = \text{LB}_\text{Keogh}_w(C,Q) \text{ if } \text{cache}_{Q,C}\text{.value }\geq d \text{ return prunedByLB} \\
\text{cache}_{Q,C} = \text{DTW}_w(C,Q) \text{ if } \text{cache}_{Q,C}\text{.value }\geq d \text{ return prunedByDTW} \\
\text{return cache}_{Q,C}\text{.value}
\]

• Next call to LazyAssessNN will be with a smaller \(w\)
• Possible to use Early Abandon on \(\text{LB}_\text{Keogh}\) and \(\text{LB}_\text{Improved}\) [1]

Fast Fill the Nearest Neighbour Table

NN. fillAll(_, ∞) ∀{w, N} \[\rightarrow\] Initialise NN table with ∞ NN distance
for s ← 2 to N do \[\rightarrow\] Start with second series
  for w ← L – 1 down to 0 do \[\rightarrow\] Start from largest window
    if NN\(^T_s\)\(_w\) ≠ ∅ then \[\rightarrow\] a. Check if NN for \(T_s\) exist at this window
      for t ← 1 to s – 1 do \[\rightarrow\] a. Update NN for all previous series
        res = LazyAssessNN\((T_s, T_t, w, NN\(^T_s\)\(_w\))\) if res not pruned then NN\(^T_s\)\(_w\) = \((T_t, \text{res})\)
      else
        for t ← 1 to s – 1 do
          res = LazyAssessNN\((T_s, T_t, w, NN\(^T_s\)\(_w\))\) if res not pruned then NN\(^T_s\)\(_w\) = \((T_t, \text{res})\)
          res = LazyAssessNN\((T_s, T_t, w, NN\(^T_t\)\(_w\))\) if res not pruned then NN\(^T_t\)\(_w\) = \((T_s, \text{res})\)
          for w' ∈ NN\(^T_s\)\(_w\).valid do NN\(^T_s\)\(_w'\) = NN\(^T_s\)\(_w\) \[\rightarrow\] d. Propagate NN for all valid windows
    b. Find NN for current series
    c. Check if current series \(T_s\) is NN for previous series
Fast Fill the Nearest Neighbour Table

• Build table for a subset $T' \subseteq T$ of increasing size until $T' = T$

1. Start with 2 time series $T' = \{T_1, T_2\}$ and fill the table as if $T'$ is the entire dataset, starting from $w = L - 1$ to $w = 0$
   • $T_2$ is the nearest neighbour of $T_1$ and vice versa

2. Add a third time series $T_3$ from $T \setminus T'$ to $T'$, $T' = \{T_1, T_2, T_3\}$
   a. Check if nearest neighbour exists for $T_3$
   b. Find the nearest neighbour of $T_3$ within $T' \setminus T_3 = \{T_1, T_2\}$
   c. Check if $T_3$ is the nearest neighbour of $T_1$ and/or $T_2$
   d. Propagate nearest neighbour of $T_3$ for all valid windows

3. Repeat step 2 with the next time series, $T_n$ in $T \setminus T'$ until $T' = T$
FastWWS Example

• Let $T$ be a training dataset of 4 time series, $T = \{T_1, T_2, T_3, T_4\}$
• Length of each time series is $L = 24$
FastWWS Example

1. Initialise **Cache** & **NN Table** with $\infty$ NN distance, NN.fillAll($-, \infty$) $\forall \{w, N\}$

2. Start with $T' = \{T_1, T_2\}$, $w = 23$, $d_{NN} = \infty$ and Query: $T_2$, Candidate: $T_1$
   - **LazyAssessNN**($T_1, T_2, 23, \infty$):
     - $\text{cache}_{T_1,T_2} = \text{LB}_\text{Kim}(T_1, T_2) = 0.040 < \infty$ continue
     - Compute $\text{cache}_{T_1,T_2} = \text{LB}_\text{Keogh}_{23}(T_1, T_2) = 0.000 < \infty$ continue
     - Compute $\text{cache}_{T_1,T_2} = \text{LB}_\text{Keogh}_{23}(T_2, T_1) = 0.046 < \infty$ continue
     - Compute $\text{cache}_{T_1,T_2} = \text{DTW}_{23}(T_1, T_2) = \{\text{validTill} = 5, 4.254\} < \infty$ **return** $\text{cache}_{T_1,T_2}.\text{value}$
     - **Assign** $T_1$ as the Nearest Neighbour for $T_2$ at $w = 23$ and vice versa for $T_1$
     - **Propagate** Nearest Neighbour of $T_2$ at $w = 23$ for $w = 22$ to 5

Reference: $\text{NN}_{w}^{T}$ (window validity, $d_{NN}$)
3. Continue with \( w = 22 \), \( d_{NN} = 4.254 \) and Query: \( T_2 \), Candidate: \( T_1 \)
   - Since we have NN for \( T_2 \) at \( w = 22 \), we have to check if \( T_2 \) is NN of \( T_1 \)
   - LazyAssessNN\((T_1, T_2, 22, \infty)\):
     - \( \text{cache}_{T_1,T_2}.\text{stoppedAt} == \text{DTW}_{23} \) and \( w = 22 \) is valid
     - \( \text{cache}_{T_1,T_2}.\text{value} = 4.254 < \infty \) return \( \text{cache}_{T_1,T_2}.\text{value} \)
   - Assign \( T_2 \) as the Nearest Neighbour for \( T_1 \) at \( w = 22 \)

4. Repeat step 3 for all windows, \( w \in \{21, \ldots, 5\} \)
5. Continue with \( w = 4, d_{\text{NN}} = \infty \) and **Query: \( T_2 \), Candidate: \( T_1 \)**
   - **LazyAssessNN** \((T_1, T_2, 4, \infty)\):
     - \( \text{cache}_{T_1,T_2} \cdot \text{stoppedAt} = \text{DTW}_5 \) and \( w = 4 \) is not valid
     - \( \text{cache}_{T_1,T_2} \cdot \text{value} = 4.254 < \infty \) **continue**
     - Compute \( \text{cache}_{T_1,T_2} = \text{LB} _{\text{Keogh}_4}(T_1,T_2) = 0.000 < \infty \) **continue**
     - Compute \( \text{cache}_{T_1,T_2} = \text{LB} _{\text{Keogh}_4}(T_2,T_1) = 2.076 < \infty \) **continue**
     - Compute \( \text{cache}_{T_1,T_2} = \text{DTW}_4(T_1,T_2) = \{\text{validTill} = 4, 4.814\} < \infty \) **return** \( \text{cache}_{T_1,T_2} \cdot \text{value} \)
   - **Assign** \( T_1 \) as the Nearest Neighbour for \( T_2 \) at \( w = 4 \) and vice versa for \( T_1 \)

6. Repeat step 5 for all windows, \( w \in \{3, \ldots, 0\} \)
### FastWWS Example

<table>
<thead>
<tr>
<th>Cache</th>
<th>StoppedAt</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$cache_{T_1,T_2}$</td>
<td>DTW$_0$</td>
<td>11.89</td>
</tr>
<tr>
<td>$cache_{T_1,T_3}$</td>
<td>LB_Kim</td>
<td>0.361</td>
</tr>
<tr>
<td>$cache_{T_2,T_3}$</td>
<td>LB_Kim</td>
<td>0.317</td>
</tr>
</tbody>
</table>

7. Add $T_3, T' = \{T_1, T_2, T_3\}$
   - $cache_{T_1,T_3} = LB_Kim(T_1, T_3) = 0.361 < \infty$
   - $cache_{T_2,T_3} = LB_Kim(T_2, T_3) = 0.317 < \infty$
   - Since $LB_Kim(T_2, T_3) < LB_Kim(T_1, T_3)$, start with $(T_2, T_3)$ pair

---

When adding a new series, initialise the row to $\infty$ - meaning no NN candidate yet
FastWWS Example

For \( T_2, T_3, \ w = 23, d_{NN} = \infty \) and Query: \( T_3 \), Candidate: \( T_2 \)

- **LazyAssessNN** \( (T_2, T_3, 23, \infty) \):
  - cache\( _{T_2,T_3} \) value = 0.317 < \( \infty \) continue
  - Compute cache\( _{T_2,T_3} = LB_{Keogh_{23}} (T_2, T_3) = 0.000 < \infty \) continue
  - Compute cache\( _{T_2,T_3} = LB_{Keogh_{23}} (T_2, T_3) = 0.000 < \infty \) continue
  - Compute cache\( _{T_2,T_3} = DTW_{23} (T_2, T_3) = \{validTill = 4, 1.612\} < \infty \) return cache. value
- Assign \( T_2 \) as the Nearest Neighbour for \( T_3 \) at \( w = 23 \)
- Since \( DTW_{23} (T_2, T_3) = 1.612 < DTW_{23} (T_1, T_2) = 4.254 \), Update \( T_3 \) as the Nearest Neighbour for \( T_2 \) at \( w = 23 \)
9. For $T_1$, $T_3$, $d_{NN} = 1.612$, $DTW_{23}(T_1, T_2) = 4.254$ and **Query**: $T_3$, **Candidate**: $T_1$

- **LazyAssessNN**($T_1, T_3, 23, 1.612$):
  - $cache_{T_1, T_3} \cdot value = 0.361 < 1.612$ **continue**
  - Compute $cache_{T_1, T_3} = LB_{Keogh}_{23}(T_1, T_3) = 0.000 < 1.612$ **continue**
  - Compute $cache_{T_1, T_3} = LB_{Keogh}_{23}(T_1, T_3) = 0.039 < 1.612$ **continue**
  - Compute $cache_{T_1, T_3} = DTW_{23}(T_1, T_3) = \{validTill = 2, 3.326\} \geq 1.612$ **return** prunedByDTW
  - No change to Nearest Neighbour for $T_3$ at $w = 23$
  - Since $DTW_{23}(T_1, T_3) = 3.326 < DTW_{23}(T_1, T_2) = 4.254$, Update $T_3$ as the Nearest Neighbour for $T_1$ at $w = 23$
10. Now we are sure about $\text{NN}_{T_{123}}^T$, $\text{NN}_{T_{223}}^T$ and $\text{NN}_{T_{323}}^T$
   
   - We can update $\text{NN}$ for $T_1, T_2, T_3$ for $w = 22$ to 4 since $\text{NN}_{T_{23}}^T$ is valid until $w = 4$
   - $\text{NN}_{T_{23}}$ is valid until $w = 2$ and will be updated later when we move on to $w = 2$
   - Since $\text{DTW}_{T_{23}}(T_2, T_3) = 1.612 < \text{DTW}_{T_{23}}(T_1, T_3) = 3.326$, start with $(T_2, T_3)$ pair for $w = 3$
   - $\text{DTW}_4(T_1, T_3) = \text{DTW}_{T_{23}}(T_1, T_3)$
   - $\text{DTW}_4(T_2, T_3) = \text{DTW}_{T_{23}}(T_2, T_3)$

### Reference:
$\text{NN}_{w}^T$ (window validity, $d_{NN}$)
11. For $T_2, T_3$ continue with $w = 3, d_{NN} = \infty$ and Query: $T_3$, Candidates: $T_2$
   - LazyAssessNN$(T_2, T_3, 3, \infty)$:
     - $\text{cache}_{T_2,T_3} \cdot \text{stoppedAt} == \text{DTW}_4$ and $w = 3$ is not valid
     - $\text{cache}_{T_2,T_3} = 1.612 < \infty$ continue
     - Compute $\text{cache}_{T_2,T_3} = \text{LB}_\text{Keogh}_3(T_2, T_3) = 0.421 < \infty$ continue
     - Compute $\text{cache}_{T_2,T_3} = \text{DTW}_3(T_2, T_3) = \{\text{validTill} = 3, 1.614\} < \infty$ return cache.value
   - Assign $T_2$ as the Nearest Neighbour for $T_3$ at $w = 3$
   - Since $\text{DTW}_3(T_2, T_3) = 1.614 < \text{DTW}_3(T_1, T_2) = 6.243$, Update $T_3$ as the Nearest Neighbour for $T_2$ at $w = 3$

12. Repeat the algorithm for all windows, $w \in \{2, \ldots, 0\}$
13. Continue adding $T_4$ to $T'$ and repeat previous steps until $T' = T = \{T_1, T_2, T_3, T_4\}$
14. Classify every instance for each window in one pass of the table
   • Yields the best window at $w = 0$ with LOO-CV accuracy of 0.75
Experimental Evaluation

• Evaluate the efficiency of FastWWS
  • LOO-CV with NN Search
    1. DTW with LB Keogh (Baseline)
    2. UCR Suite
    3. Pruned DTW with LB Keogh
    4. UCR Suite with Pruned DTW
  • LOO-CV with FastWWS

• Exhaustive search on all methods

• Average results over 10 runs for different reshuffling of \( T \)

• 85 benchmark time series datasets
  http://www.cs.ucr.edu/~eamonn/time_series_data/

```plaintext
for w = 0 to L do
  error = 0
  for each s in T do
    nn_s = nn_search(s, T-s, w)
    if nn_s.class \neq s.class then error++
  if error < bestError then
    bestWW = w
    bestError = error
```
FastWWS is **FASTER** and more **EFFICIENT** than all known methods!

State of the arts: 10s

FastWWS: 1s

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HandOutlines ($L = 2709$) - 1000x</td>
<td>105</td>
</tr>
<tr>
<td>DTW with LB Keogh</td>
<td>100</td>
</tr>
<tr>
<td>DTW with UCR Suite</td>
<td></td>
</tr>
<tr>
<td>PrunedDTW with LB Keogh</td>
<td>15</td>
</tr>
<tr>
<td>PrunedDTW with UCR Suite</td>
<td>14</td>
</tr>
<tr>
<td>FastWWS</td>
<td>2.5 hours</td>
</tr>
</tbody>
</table>

**Average speed up**

On the $N^2$ Term

- $N \leq 200 \Rightarrow 106x$
- $N > 200 \Rightarrow 184x$

On the $L^3$ Term

- $L \leq 300 \Rightarrow 67x$
- $L > 300 \Rightarrow 250x$
FastWWS can **SCALE** too!

At just above **20k**, LB Keogh takes more than a day

At around **45k**, UCR Suite takes more than a day

More than a week at **100k**

FastWWS takes only **6 hours**

The short length \( (L = 24) \) affects PrunedDTW
FastWWS with PrunedDTW

1. Compute Euclidean Distance ($w = 0$)
2. Use it as upper bound to prune DTW at larger window

- Not necessary faster
- **FastWWS** is faster on 55% of the Benchmark datasets
- Due to overhead in **PrunedDTW** in checking the upper bounds
Classification Accuracy

Accuracy should be the same as the window found is the same and FastWWS is EXACT

<table>
<thead>
<tr>
<th>Datasets</th>
<th>LB_Keogh</th>
<th>UCR Suite</th>
<th>LB_Keogh PrunedDTW</th>
<th>UCR Suite PrunedDTW</th>
<th>Fast WWS Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>50Words</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Adiac</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>ArrowHead</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Beef</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BeetleFly</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>BirdChicken</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>CBF</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Car</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>ChlorineConcentration</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CinCECG_torso</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Coffee</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Computers</td>
<td>74</td>
<td>74</td>
<td>74</td>
<td>74</td>
<td>74</td>
</tr>
<tr>
<td>Cricket_X</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>Cricket_Y</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>Cricket_Z</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>DistortionSizeReduction</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DistalPhalanxOutlineAgeGroup</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DistalPhalanxOutlineCorrect</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>DistalPhalanxTW</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ECG2000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ECG5000</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ECGFiveDays</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Earthquakes</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>ElectricDevices</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>FISH</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>FaceAll</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>FaceFour</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>FaceSCU</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>FordA</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>FordB</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>
Conclusions

• A novel and exact algorithm to speed up the search for the best parameter (warping window) for DTW
  • FastWWS is more EFFICIENT and FASTER
  • FastWWS can SCALE

• Our results, datasets and source code are online at
  • https://bit.ly/SDM18
  • https://github.com/ChangWeiTan/FastWWSearch
  • Slides: http://changweitan.com/research/SDM18-slides.pdf
Future Work

• Search for the best parameter for other TS similarity functions
  • LCSS ($\delta, \varepsilon$), MSM ($c$), ERP ($g, \lambda$) etc.
  • Satisfies the three properties:
    1. Its **distance** stays **valid** for some parameters
    2. Its **distance** is **monotone** with its parameters
    3. Its **lower bound** is **monotone** with its parameters

• Scaling up the State of the Arts in Time Series Classification
  • Elastic Ensembles (EE) [1]
  • Collective of Transformation-Based Ensembles (COTE) [2]

Thank you!

Questions and Answers

CW. Tan  M. Herrmann  G. Forestier  G.I. Webb  F. Petitjean

This work was supported by the Australian Research Council under grant DE170100037. This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-16-1-4023

chang.tan@monash.edu  github.com/ChangWeiTan/FastWWSearch  bit.ly/SDM18