Indexing and Classifying Gigabytes of Time Series under Time Warping

C.W. Tan
G.I. Webb
F. Petitjean

2017 SIAM International Conference on DATA MINING
27 April 2017
Temporal Land-Cover Maps
What can we do with it?

• Yield forecast
What can we do with it?

- Yield forecast
- Fire spread model
What can we do with it?

- Yield forecast
- Fire spread model
- City pollution absorption models
- and more...
One Image is not enough!

Impossible to differentiate them!
What’s possible? → Temporal Evolution

Satellite Image Time Series (SITS) Analysis

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

How to do this?

• Time series classification

• State-of-the-art, Nearest Neighbor coupled with Dynamic Time Warping (NN-DTW) [1]
  • Many phenomena of interest – vegetation cycles, have periodic behavior which can be modulated by weather artifacts. [2]
  • Too short for the Bag-of-word-type approaches to perform best
    • Length of 46 – 52
    • Less features in the series
    • BOSS-VS [3] achieved around 40% error rate, NN-DTW achieved 16%

Example series for different crops

- Corn
- Soybean
- Wheat
- Broad-Leaved Tree
Traditionally

\[ \text{A million pixels} = \text{A million sequences} \]

\[ \text{X 1,000,000} \]

\[ \text{100 million examples} \]

How long will it take?
Most research in time series classification

Average NN-DTW Classification Time

- 7 billion SITS, size of Texas: 30k years
- 16 million SITS, size of Houston: 1 year
- 1 million SITS: 10 days
- UCR Time Series Archive: < 10,000 (under 30 mins)

Most research in time series classification
Problem Statement

• **Anytime Time Series Classification**
  • Classify a query at any given time with high accuracy
  • Without constraints on computational resources at training time

• **In Nearest Neighbor classification**
  • Find the nearest neighbor much faster than full linear scan
  • Traditional techniques
    • Build an indexing structure in Euclidean Space
    • k-d tree, R tree, LSH ...
    • Does not work with DTW
Indexing with Hierarchical Clusters

A (9) → Root
D (9)  E (9)  F (7)

B (1) → Root
G (1)  H (14)  I (1)

C (2) → Root
J (2)  K (2)  L (14)
Time Series Indexing

- Hierarchical K-means indexing structure
  - Uses a priority search to speedup the process [1]
- Leverage off a recent work on DTW averaging
  - DTW Barycenter Averaging (DBA) [2, 3]
  - [2] shows that K-means and DBA allows faster and more accurate classification

Time Series Indexing

- At testing time

\[ \text{SearchTree}(T, Q, K) \]

\[ \text{PQ, Res} = \text{empty priority queues} \]

\[ \text{Traverse}(T, Q, PQ, Res) \]

\[ \text{while (within contract and PQ not empty) do} \]

\[ \text{nextBranch} = \text{PQ.pop()} \]

\[ \text{Traverse}(\text{nextBranch}, Q, PQ, Res) \]

\[ \text{end while} \]

\[ \text{return Res.pop}(k) \]

\[ \text{Traverse}(T, Q, PQ, Res) \]

\[ \text{if (T is leaf) then} \]

\[ \text{Res.addAll}(T.data) \text{ with distances to Q} \]

\[ \text{else} \]

\[ C = T.child \text{ nearest to } Q \]

\[ \text{PQ.addAll}(T.child \text{ except } C) \text{ with distances to Q} \]

\[ \text{Traverse}(C, Q, PQ, Res) \]

\[ \text{end if} \]
Time Series Indexing

- At testing time

\[ \text{SearchTree}(T, Q, K) \]

\[ PQ, Res = \text{empty priority queues} \]

\[ \text{Traverse}(T, Q, PQ, Res) \]

\[ \text{while} \ (\text{not stop and PQ not empty}) \ \text{do} \]

\[ \text{nextBranch} = \text{PQ}.\text{pop}() \]

\[ \text{Traverse}(\text{nextBranch}, Q, PQ, Res) \]

\[ \text{end while} \]

\[ \text{return Res.pop}(k) \]

\[ \text{Traverse}(T, Q, PQ, Res) \]

\[ \text{if} \ (T \text{ is leaf}) \ \text{then} \]

\[ \text{Res.addAll}(T.\text{data}) \text{ with distances to } Q \]

\[ \text{else} \]

\[ C = T.\text{child nearest to } Q \]

\[ \text{PQ.addAll}(T.\text{child except } C) \text{ with distances to } Q \]

\[ \text{Traverse}(C, Q, PQ, Res) \]

\[ \text{end if} \]

Apply DTW lower bounds, \textit{LB Keogh} to minimize DTW computations and have 2 PQ

These are a NN search with DTW \( O(L^2) \) time
Lower Bound Keogh (LB Keogh)

1. Computes **Upper** (U) and **Lower** (L) envelope for query Q
2. Computes the distance of the projection of a candidate sequence C onto the envelope

Only need to compute the envelopes for Q once!!

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


Simple example
Time Series Indexing Example

- **Alphabets** are Centroids of each cluster
- **Numbers** are actual time series in training set
- 23 time series in the training set

Classes: Blue Red

```
Classes: Blue Red
```

```
Alphabets are Centroids of each cluster
Numbers are actual time series in training set
23 time series in the training set
```
Time Series Indexing Example

Query time series
Actual NN: 13
Time Series Indexing Example

Query time series
Actual NN: 13

LB Distance to
A: 0.895
B: 6.157
C: 0.814

DTW Distance to
A: 4.893
B: Skip (16.920)
C: 5.231

LB Priority Queue : {B}
Priority Queue Distance to Query : {6.2}
DTW Priority Queue : {C}
Priority Queue Distance to Query : {5.2}
Time Series Indexing Example

Query time series
Actual NN: 13

LB Distance to
6: 20.253
D: 0.573
2: 0.781

DTW Distance to
6: Skip (40.592)
D: 6.668
2: 10.194

LB Priority Queue: {B, 6}
Priority Queue Distance to Query: {6.2, 20.3}
DTW Priority Queue: {C, 2}
Priority Queue Distance to Query: {5.2, 10.2}
Time Series Indexing Example

Query time series
Actual NN: 13

LB Distance to
H: 1.252
I: 0.726
19: 1.321

DTW Distance to
H: 11.387
I: 4.839
19: 9.335

Target

LB Priority Queue : {B, 6}
Priority Queue Distance to Query : {6.2, 20.3}
DTW Priority Queue : {C, 19, H, 2}
Priority Queue Distance to Query : {5.2, 9.3, 11.4, 10.2}
Time Series Indexing Example

Query time series
Actual NN: 13

LB Distance to
18: 1.097
21: 1.726

DTW Distance to
18: 4.911
21: 9.548

LB Priority Queue
{B, 6}
Priority Queue Distance to Query
{6.2, 20.3}
DTW Priority Queue
{C, 19, H, 2}
Priority Queue Distance to Query
{5.2, 9.3, 11.4, 10.2}
Time Series Indexing Example

Query time series
Actual NN: 13

• Current NN is 18, Class 1
• Not actual NN
• Next to explore is Node C
• Dequeue C from DTW Priority Queue

Next to explore
LB Distance of B > DTW Distance of C

NN : {18}
Distance to Query : 4.911

LB Priority Queue : {B, 6}
Priority Queue Distance to Query : {6.2, 20.3}
DTW Priority Queue : {C, 19, H, 2}
Priority Queue Distance to Query : {5.2, 9.3, 11.4, 10.2}
Time Series Indexing Example

Query time series
Actual NN: 13

LB Distance to
13: 0.672
F: 0.497
G: 2.585

DTW Distance to
13: 2.930
F: 4.249
G: 11.446

Target
Actual NN: 13
Distance to Query: 2.930

LB Priority Queue: {B, 6}
Priority Queue Distance to Query: {6.2, 20.3}
DTW Priority Queue: {F, 19, H, 2, G}
Priority Queue Distance to Query: {4.2, 9.3, 11.4, 10.2, 11.4}
Time Series Indexing Example

Query time series
Actual NN: 13

- Found NN in 2 tree traversals
- Next to explore is Node F
- Dequeue F from DTW Priority Queue

LB Priority Queue: \{B, 6\}
Priority Queue Distance to Query: \{6.2, 20.3\}
DTW Priority Queue: \{F, 19, H, 2, G\}
Priority Queue Distance to Query: \{4.2, 9.3, 11.4, 10.2, 11.4\}

Target

NN: \{13\}
Distance to Query: 4.249

Next to explore
LB Distance of B > DTW Distance of F
Comparison with state of the art
Experiments

- Compared with NN-DTW with LB_Keogh
  - at x % of the time of the full NN-DTW
  - 1%, 10%, 20%, 30%, 40%, 50%

- Satellite Dataset
  - Train 1M series
  - Length 46
  - Number of classes: 24

- 84 UCR Repository [1]

Results on the satellite data

State of the art – random sampling

Our approach

If given only 0.1ms to classify a pixel, we do better by 22%

At 1ms to classify a pixel, we do better by 18%

Almost same accuracy as full search but 1,000x faster!
• Classifying Houston would take 4 hours instead of 1 year!
Performance on UCR repository

- Look at how well we perform if we are given $x\%$ of the time of the full NN-DTW.

<table>
<thead>
<tr>
<th>Intervals</th>
<th>LB_Keogh NN-DTW</th>
<th>TSI</th>
<th>$R^+$</th>
<th>$R^-$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>1.529</td>
<td>1.471</td>
<td>2034.5</td>
<td>1620.5</td>
<td>-0.907</td>
</tr>
<tr>
<td>10%</td>
<td>1.841</td>
<td>1.159</td>
<td>3449</td>
<td>206</td>
<td>-7.105</td>
</tr>
<tr>
<td>20%</td>
<td>1.871</td>
<td>1.129</td>
<td>3451</td>
<td>204</td>
<td>-7.114</td>
</tr>
<tr>
<td>30%</td>
<td>1.806</td>
<td>1.194</td>
<td>3219.5</td>
<td>435.5</td>
<td>-6.099</td>
</tr>
<tr>
<td>40%</td>
<td>1.741</td>
<td>1.259</td>
<td>2903</td>
<td>752</td>
<td>-4.713</td>
</tr>
<tr>
<td>50%</td>
<td>1.671</td>
<td>1.329</td>
<td>2616</td>
<td>1039</td>
<td>-3.455</td>
</tr>
<tr>
<td>Average</td>
<td>1.743</td>
<td>1.257</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Our results, datasets and source code are online at
https://github.com/ChangWeiTan/TSI
Future Work

• Pruning the whole branch
  • Atomic Wedgie [1]
  • If everything in that branch is of the same class

• Optimizing the branching factor, $K$
  • Vary $K$ and keep the $K$ value that gives the best trade-off between query time and error rate.

• Speeding up search for the best warping window on large dataset
  • Current method via Cross Validation

Take home message

1. The first algorithm (TSI) to index DTW-induced space
   - Hierarchical K-means tree
   - DTW Barycenter Averaging (DBA)

2. Twice the accuracy than NN-DTW on large (1M) remote sensing data if given 1ms to classify a query

3. Perform better even on smaller datasets
   - If we just have 50% of the full search time.
Thank you!

Questions and Answers

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. This work was supported by the Australian Research Council under awards DE170100037 and DP140100087, and by the 2016 IBM Faculty Award (F. Petitjean).
Additional Information
Lower Bound for DTW

**LinearScan(Q)**

- bestSoFar = infinity
- for each sequence S in database
  - dtwDist = DTW(Q, S)
  - if (dtwDist < bestSoFar) then
    - bestSoFar = dtwDist
    - nn = S
  - end if
- end for
- return nn

**LowerBoundScan(Q)**

- bestSoFar = infinity
- for each sequence S in database
  - lbDist = LowerBound(Q, S)
  - if (lbDist < bestSoFar) then
    - dtwDist = DTW(Q, S)
    - if (dtwDist < bestSoFar) then
      - bestSoFar = dtwDist
      - nn = S
    - end if
  - end if
- end for
- return nn

Cheap test before computing the actual DTW distance
Time Series Indexing

- At training time

\textbf{BuildTree}(data, K)

\begin{align*}
\text{if } (|data| \leq K) & \text{ then} \\
& \text{create leaf node with all the data} \\
\text{else} & \\
(C,P) & = \text{Kmeans}(data,K) \\
\textbf{for each} \text{ cluster } C_i \text{ do} & \\
& \text{create node } N_i = \text{BuildTree}(C_i, K) \\
& \text{assign center } P_i \text{ to } N_i \\
\textbf{end for} & \\
\text{end if}
\end{align*}

Replace arithmetic mean with DBA
Example on SITS Dataset
Example 2
Time Series Indexing Example

Query time series
Actual NN: 11

LB Distance to
A: 2.990
B: 10.900
C: 0.302

DTW Distance to
A: Skip (2.917)
B: Skip (5.348)
C: 1.316

LB Priority Queue : {A, B}
Priority Queue Distance to Query : {3.0, 10.9}
DTW Priority Queue : {}
Priority Queue Distance to Query : {}
Time Series Indexing Example

Query time series
Actual NN: 11

LB Distance to
13: 4.087
F: 1.876
G: 0.047

DTW Distance to
13: Skip (2.536)
F: Skip (2.592)
G: 0.9998

LB Priority Queue : {F, A, 13, B}
Priority Queue Distance to Query : {1.9, 3.0, 4.1, 10.9}
DTW Priority Queue : {}
Priority Queue Distance to Query : {}
Time Series Indexing Example

Query time series
Actual NN: 11

LB Distance to
K: 0.059
L: 0.225
M: 0.226

DTW Distance to
K: 0.281
L: 2.913
M: 3.791

LB Priority Queue: \{F, A, 13, B\}
Priority Queue Distance to Query: \{1.9, 3.0, 4.1, 10.9\}
DTW Priority Queue: \{L, M\}
Priority Queue Distance to Query: \{2.9, 3.8\}
Time Series Indexing Example

Query time series
Actual NN: 11

LB Distance to
1: 0.063
11: 0.064

DTW Distance to
1: 0.508
11: 0.207

LB Priority Queue : {F, A, 13, B}
Priority Queue Distance to Query : {1.9, 3.0, 4.1, 10.9}
DTW Priority Queue : {L, M}
Priority Queue Distance to Query : {2.9, 3.8}

KNN Priority Queue : {11}
Distance to Query : 0.207
Time Series Indexing Example

Query time series
Actual NN: 11

- Found NN in 1 tree traversal
- Assign to class 1
- Next to be explore is node L or F
- Can stop here or until contract exhausted

LB Priority Queue: \{F, A, 13, B\}
Priority Queue Distance to Query: \{1.9, 3.0, 4.1, 10.9\}
DTW Priority Queue: \{L, M\}
Priority Queue Distance to Query: \{2.9, 3.8\}

KNN Priority Queue: \{11\}
Distance to Query: 0.207

Next to explore
LB Distance of F < DTW Distance of L

Target