CCE Dept. Colloquium

Support Vector Machines and Kernels on Time-Series

mcuturi@i.kyoto-u.ac.jp
Outline of the Talk

Very brief introduction to the Support Vector Machine

- Intuition and computation
- Geometric interpretation

Very brief introduction to kernel methods

- What are kernels in a machine learning context?

Present new work on kernels for time-series

- Inspired by the Dynamic Time Warping Distance
  - Cuturi, *Fast Global Alignment Kernels* (ICML 2011)
Support Vector Machines
Binary Classification Separation Surfaces for Vectors

What is a classification rule?
Classification rule = a partition of $\mathbb{R}^d$ into two sets
Classification Separation Surfaces for Vectors

Can be defined by a single surface, \textit{e.g.} a curved line
Even more simple: using straight lines and halfspaces.
Classification Separation Surfaces for Vectors

Given two sets of points...

Some slides from now on are taken from Jean-Philippe Vert's lectures
Classification Separation Surfaces for Vectors

It is sometimes possible to separate them perfectly
Classification Separation Surfaces for Vectors

Each choice might look equivalently good on the training set, but it will have obvious impact on new points.
Classification Separation Surfaces for Vectors
Linear classifier, some degrees of freedom
Linear classifier, some degrees of freedom
Linear classifier, some degrees of freedom

Specially close to the border of the classifier
Linear classifier, some degrees of freedom
Linear classifier, some degrees of freedom

For each different technique, different results, different performance.
A criterion to select a linear classifier: the margin

Idea: look for the biggest possible “buffer” between red and blue points.
A criterion to select a linear classifier: the margin
A criterion to select a linear classifier: the margin
A criterion to select a linear classifier: the margin
A criterion to select a linear classifier: the margin
Largest Margin Linear Classifier?
Support Vectors with Large Margin
Finding the optimal hyperplane

- Consider \( n \) labeled points \((x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}\), with \( i = 1, \ldots, n \).
- Finding the optimal hyperplane is equivalent to finding \((w, b)\) which minimize:

\[
\|w\|^2
\]

under the constraints:

\[
\forall i = 1, \ldots, n, \quad y_i (w^T x_i + b) - 1 \geq 0.
\]

This is a classical quadratic program on \( \mathbb{R}^{d+1} \), with linear constraints - quadratic objective.
Dual problem

• introduce one dual variable $\alpha_i$ for each constraint,

The dual problem is

$$
\begin{align*}
\text{maximize} & \quad g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\text{such that} & \quad 0 \leq \alpha_i, \sum_{i=1}^{n} \alpha_i y_i = 0.
\end{align*}
$$

This is a quadratic program in $\mathbb{R}^n$, with box constraints. $\alpha^*$ can be computed using elementary optimization software (e.g. built-in matlab function).

• Strong duality holds. KKT gives us $\alpha_i (y_i (w^T x_i + b) - 1) = 0$,

...hence, either $\alpha_i = 0$ or $y_i (w^T x_i + b) = 1$.

• $\alpha_i \neq 0$ only for points on the support hyperplanes $\{(x, y) | y_i(w^T x_i + b) = 1\}$. 
The final solution

- With $\alpha^*$, we can recover $(w^*, b^*)$.
- The **decision function** is therefore:
  
  $$f^*(x) = (w^*)_T x + b^*$$

  $$= \left( \sum_{i=1}^{n} y_i \alpha_i x_i^T \right) x + b^*.$$  

- Here the **dual** solution gives us directly the **primal** solution.
Interpretation: support vectors

\[ \alpha > 0 \]

\[ \alpha = 0 \]
Another interpretation: Convex Hulls

go back to 2 sets of points that are linearly separable
Another interpretation: Convex Hulls

Linearly separable = convex hulls do not intersect
Another interpretation: Convex Hulls

Find two closest points, one in each convex hull
Another interpretation: Convex Hulls

The SVM = bisection of that segment
Another interpretation: Convex Hulls

support vectors = extreme points of the faces on which the two points lie
The non-linearly separable case

(when convex hulls intersect)
What happens when the data is not linearly separable?
What happens when the data is not linearly separable?
What happens when the data is not linearly separable?
What happens when the data is not linearly separable?
Soft-margin SVM?

- Find a trade-off between **large margin** and **few errors**.

- Mathematically:

\[
\min_f \left\{ \frac{1}{\text{margin}(f)} + C \times \text{errors}(f) \right\}
\]

- \( C \) is a parameter
Dual formulation of soft-margin SVM

The dual program corresponding to this “softer” formulation is

\[
\begin{align*}
\text{maximize} \quad & g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\text{such that} \quad & 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \ldots, n, \\
& \sum_{i=1}^{n} \alpha_i y_i = 0.
\end{align*}
\]
Interpretation: bounded and unbounded support vectors

- $\alpha = 0$
- $0 < \alpha < C$
- $\alpha = C$
What about the convex hull analogy?

- Remember the separable case

- Here we consider the case where the two sets are not linearly separable, i.e. their convex hulls intersect.
What about the convex hull analogy?

**Definition 1.** Given a set of \( n \) points \( A \), and \( 0 \leq C \leq 1 \), the set of finite combinations

\[
\sum_{i=1}^{n} \lambda_i x_i, 1 \leq \lambda_i \leq C, \sum_{i=1}^{n} \lambda_i = 1,
\]

is the \( (C) \) reduced convex hull of \( A \)

- Using \( C = 1/2 \), the reduced convex hulls of \( A \) and \( B \),

- Soft-SVM with \( C = \) closest two points of \( C \)-reduced convex hulls.

Images taken from "Duality and geometry in SVM classifiers," Bennett and Bredensteiner
The Kernel Trick in SVM’s
Kernel trick for SVM’s

- use a mapping $\phi$ from $\mathcal{X}$ to a feature space,
- which corresponds to the kernel $k$:

$$\forall x, x' \in \mathcal{X}, \quad k(x, x') = \langle \phi(x), \phi(x') \rangle$$

- Example: if $\phi(x) = \phi \left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix}$, then

$$k(x, x') = \langle \phi(x), \phi(x') \rangle = (x_1)^2(x'_1)^2 + (x_2)^2(x'_2)^2.$$
Training a SVM in the feature space

Replace each $x^T x'$ in the SVM algorithm by $\langle \phi(x), \phi(x') \rangle = k(x, x')$

- The dual problem becomes

$$g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j),$$

under the constraints:

$$\begin{cases} 
0 \leq \alpha_i \leq C, & \text{for } i = 1, \ldots, n \\
\sum_{i=1}^{n} \alpha_i y_i = 0.
\end{cases}$$

- The decision function becomes:

$$f(x) = \langle w, \phi(x) \rangle + b^*$$

$$= \sum_{i=1}^{n} y_i \alpha_i k(x_i, x) + b^*. \quad (1)$$
The explicit computation of $\phi(x)$ is not necessary. The kernel $k(x, x')$ is enough.

- the SVM optimization for $\alpha$ works **implicitly** in the feature space.
- the SVM is a kernel algorithm: only need to input $K$ and $y$:

  \[
  \begin{align*}
  \text{maximize} \quad & g(\alpha) = \alpha^T \mathbf{1} - \frac{1}{2} \alpha^T (K \odot yy^T) \alpha \\
  \text{such that} \quad & 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \ldots, n \\
  & \sum_{i=1}^{n} \alpha_i y_i = 0.
  \end{align*}
  \]

- $K$’s **positive definite** $\Rightarrow K \odot yy^T \Leftrightarrow$ problem is convex
- the decision function is $f(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot) + b.$
Kernel example: polynomial kernel

• For $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$, let $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \in \mathbb{R}^3$:

$$K(x, x') = x_1^2x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2x_2'^2$$

$$= (x_1x_1' + x_2x_2')^2$$

$$= (x^T x')^2.$$
Kernels are Trojan Horses onto Linear Models

- With kernels, complex structures can enter the realm of linear models
A few words about Kernel Methods
Kernel Methods

- Popular in machine learning now

- Gained momentum in the late 90’s with the support vector machine,

- Cross-disciplinary: Statistics, Optimization, Functional Analysis, Linear Algebra
A kernel on a set $\mathcal{X}$ is...

any function

$$k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$$

$$(x, y) \rightarrow k(x, y),$$

which is symmetric

$$k(x, y) = k(y, x),$$

and positive-definite:

for any family of points $x_1, \cdots, x_n$ of $\mathcal{X}$, the matrix

$$K = \begin{bmatrix}
k(x_1, x_1) & \cdots & k(x_1, x_i) & \cdots & k(x_1, x_n) \\
\vdots & \ddots & \vdots & & \vdots \\
k(x_i, x_1) & \cdots & k(x_i, x_i) & \cdots & k(x_i, x_n) \\
\vdots & \ddots & \vdots & & \vdots \\
k(x_n, x_1) & \cdots & k(x_n, x_i) & \cdots & k(x_n, x_n)
\end{bmatrix} \succeq 0$$

is positive (semi)definite (has nonnegative eigenvalues).
The general framework of kernel methods

\[ f(x) = \sum_{i=1}^{N} \alpha_i k(x_i, x) \]

weights \( \alpha \) estimated with a kernel machine

Kernel methods optimize weights \( \alpha \) to avoid overfitting & improve performance by using convex optimization
Positive Definiteness of $K \Rightarrow$ Convex Optimization

!! Important remark !!

convex optimization only works because the kernel is positive definite
Kernels for Time Series
very Few Kernels on Time Series

Kernels for structured objects

- Large literature:
  - Kernels for images,
  - Kernels for graphs,
  - Kernels for histograms, Bags-of-Words representations
  - Kernels for sequences: DNA, proteins: discrete symbols.

What about time-series?

- **Important task**: Ubiquitous in science and engineering
- **Room for improvement**: very few proposals in literature so far
Time-series: a collection of objects indexed by time

- Images
• **Univariate** time-series (google stock on a day)

![Chart of Google stock over a day](chart.png)

• **Multivariate** time-series (compiled by monks in 11th century!)

![Multivariate time-series chart](chart.png)

**Objective:** propose positive definite kernels between two time-series $\mathbf{x} = (x_1, \cdots, x_n)$ $\mathbf{y} = (y_1, \cdots, y_m)$ where the $x_i, y_j$ belong to the same arbitrary set $\mathcal{X}$
Measuring similarity between time-series

- Time-series look like vectors, yet, in most cases:
  - neighboring coefficients $x_i$ and $x_{i+1}$ are not independent (smoothness)
  - causality: early observations $x_i$ condition ulterior observations $x_{i+...}$
  - time-series in a dataset have different lengths.

- Even if we assume $n = m$, the Euclidean distance is blind to these subtleties:

\[
d_{\text{Euclidean}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} d(x_i, y_i).
\]
Dynamic Time Warping (1971)

- First proposed by Japanese researchers in Japan: H. Sakoe & S. Chiba
- **Huge impact** in engineering: first in speech, now all domains of science
- **idea**: find a **good alignment** between $x$ and $y$ before computing $d_{\text{Euclidean}}$.

$$d_{\text{DTW}}(x, y) = \min_{\pi \in \mathcal{A}(x,y)} \sum_{i=1} d(x_{\pi_1(i)}, y_{\pi_2(i)}) = \min_{\pi \in \mathcal{A}(x,y)} d_{\text{Euclidean}}(x_{\pi_1}, y_{\pi_2})$$

image taken from http://www.markcorbyn.com
Alignments

- Here are two sequence aligned

- An alignment is an increasing path on a grid.
We first “lay out” the $n \times m$ grid, corresponding to $\mathbf{x} = (x_1, \cdots, x_5)$ $\mathbf{y} = (y_1, \cdots, y_7)$
Optimal Alignment

The grid is filled with pairwise distances.
Optimal Alignment

This rectangular matrix is the only thing we need.
Optimal Alignment

An alignment is a path that starts from \((1, 1)\) to reach \((5, 7)\)
Optimal Alignment

The only admissible moves from one cell to the next are →, ↑ and ↗
The cost of a path is the sum of contributions $D_{ij}$ it walks through.
So far,

\[ C = D_{11}. \]
Optimal Alignment

Moving up,

\[ C = D_{11} + D_{21}. \]
Moving diagonally,

\[ C = D_{11} + D_{21} + D_{32}. \]
Moving right,

\[
C = D_{11} + D_{21} + D_{32} + D_{33}.
\]
Optimal Alignment

\[ C = D_{11} + D_{21} + D_{32} + D_{33} + D_{34} + D_{35} + D_{45} + D_{46} + D_{57}. \]

\( \text{etc.}, \) until we reach the upper right corner.
A path is uniquely defined by 2 rows vectors:

\[ C = D_{11} + D_{21} + D_{32} + D_{33} + D_{34} + D_{35} + D_{45} + D_{46} + D_{57}. \]
A path is uniquely defined by 2 rows vectors:

$$\pi = \begin{pmatrix} \pi_1 \\ \pi_2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 3 & 3 & 3 & 4 & 4 & 5 \\ 1 & 1 & 2 & 3 & 4 & 5 & 5 & 6 & 7 \end{pmatrix}$$
Given a path $\pi$, we call $C(\pi)$ the sum of distances:

$$C(\pi) = D_{11} + D_{21} + D_{32} + D_{33} + D_{34} + D_{35} + D_{45} + D_{46} + D_{57}.$$
Optimal Alignment

We defined the distance $d_{DTW}$ as

$$d_{DTW}(x, y) = \min_{\pi \in A(x, y)} \sum_{i=1}^{\vert p_i \vert} d(\pi_1(i), \pi_2(i)) = \min_{\pi \in A(x, y)} C_{x, y}(\pi).$$
\( \mathcal{A}(x, y) \Leftrightarrow \) the set of all paths on this grid. Only depends on the \(|x| \textbf{ and } |y|\), 5 and 7 here.
To clarify this, we write $\mathcal{A}(|x|, |y|)$ for the set of all alignments between $x$ and $y$. 
Optimal Alignment

card $A(n, m)$ is equal to the **Delannoy number** $\text{Delannoy}(n, m)$. 
Optimal Alignment

\[
\begin{array}{cccccccc}
  x_5 & D_{51} & D_{52} & D_{53} & D_{54} & D_{55} & D_{56} & D_{57} \\
  x_4 & D_{41} & D_{42} & D_{43} & D_{44} & D_{45} & & & \\
  x_3 & & D_{31} & D_{32} & D_{33} & D_{34} & D_{35} & D_{36} & D_{37} \\
  x_2 & & & D_{21} & D_{22} & D_{23} & D_{24} & D_{25} & D_{26} \\
  x_1 & & & & D_{11} & D_{12} & D_{13} & D_{14} & D_{15} \end{array}
\]

\[\text{Delannoy}(5, 7) = 2241\]
\[\vdots\]
\[\text{Delannoy}(20, 20) = 4.53e + 13\]
DTW finds the minimum among all paths: *discrete optimization*. Obviously, checking each would be *computationally intractable*.
Optimal Alignment

Key idea: use **Bellman’s Dynamic programming**
Define $C^*_{i,j}$ as the **cost of the optimal sub-path** up to the $i$-th symbol of $x$ and the $j$-th symbol of $y$.

$$C^*_{i,j} = \min_{\pi \in A(i,j)} C_{x_i, y_j}^\pi(\pi).$$
Obviously $C_{57}^*$ is the quantity we want to compute.
**Optimal Alignment**

<table>
<thead>
<tr>
<th></th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_{34}$</td>
<td>$D_{35}$</td>
<td>$D_{56}$</td>
<td>$C^*_57$</td>
</tr>
<tr>
<td>$D_{44}$</td>
<td>$D_{45}$</td>
<td>$C^*_46$</td>
<td>$C^*_47$</td>
</tr>
<tr>
<td>$D_{54}$</td>
<td>$D_{55}$</td>
<td>$C^*_56$</td>
<td></td>
</tr>
</tbody>
</table>

Relationship between $C^*_57$ its neighbours $C^*_56$, $C^*_46$, $C^*_47$?
### Optimal Alignment

\[
\begin{array}{cccc}
  x_5 & D_{54} & D_{55} & C_{56}^* & C_{57}^* \\
  x_4 & D_{44} & D_{45} & C_{46}^* & C_{47}^* \\
  x_3 & D_{34} & D_{35} & D_{36} & D_{37} \\
  y_4 & & & & \\
  y_5 & & & & \\
  y_6 & & & & \\
  y_7 & & & & \\
\end{array}
\]

\[
C_{57}^* = \min(C_{56}^*, C_{46}^*, C_{47}^*) + D_{57}
\]
Optimal Alignment

More generally, for all \(i \leq n - 1, j \leq m - 1\),

\[
C^{*}_{i+1,j+1} = \min(C^{*}_{i+1,j}, C^{*}_{i,j}, C^{*}_{i,j+1}) + D_{i+1,j+1}
\]
We first compute $C_{1,1}^*$.
Easy, since $C_{1,1}^\ast = D_{1,1}$
We now compute $C_{2,1}^* = C_{1,1}^* + D_{2,1}$.
Optimal Alignment

Same for $C_{3,1}^*$...
Optimal Alignment

\[ D_{41}, C_{41}^*, C_{31}^*, C_{21}^*, C_{11}^* \]

\[ \ldots C_{4,1}^* \ldots \]
Optimal Alignment

\[ D_{51} \]
\[ C_{51}^* \]
\[ D_{41} \]
\[ C_{41}^* \]
\[ D_{31} \]
\[ C_{31}^* \]
\[ D_{21} \]
\[ C_{21}^* \]
\[ D_{11} \]
\[ C_{11}^* \]

... and \( C_{5,1}^* \)...
Optimal Alignment

\[
\begin{array}{cccccccc}
\cdots & C_{51} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & C_{41} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & C_{31} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & C_{21} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & C_{11} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{array}
\]

\( C_{1,2}^* \) depends only on \( C_{1,1}^* \) and \( D_{1,2} \)
We now apply Bellman's recurrence for the first time:

\[ C_{22}^* = \min(C_{21}^*, C_{11}^*, C_{12}^*) + D_{22} \]
## Optimal Alignment

![Optimal Alignment Diagram](image)

- \(x_1\)
  - \(y_1\): \(C_{11}\)
  - \(y_2\): \(C_{12}\)

- \(x_2\)
  - \(y_3\): \(D_{21}\)
  - \(y_4\): \(C_{22}\)

- \(x_3\)
  - \(y_5\): \(D_{31}\)
  - \(y_6\): \(C_{32}\)

- \(x_4\)
  - \(y_7\): \(D_{41}\)

- \(x_5\)
  - \(y_8\): \(C_{51}\)

...
Optimal Alignment

<table>
<thead>
<tr>
<th>x5</th>
<th>(C^*_5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D51</td>
<td>(C^*_1)</td>
</tr>
<tr>
<td>x4</td>
<td>(C^*_4)</td>
</tr>
<tr>
<td>D41</td>
<td>(C^*_2)</td>
</tr>
<tr>
<td>x3</td>
<td>(C^*_3)</td>
</tr>
<tr>
<td>D31</td>
<td>(C^*_3)</td>
</tr>
<tr>
<td>x2</td>
<td>(C^*_2)</td>
</tr>
<tr>
<td>D21</td>
<td>(C^*_2)</td>
</tr>
<tr>
<td>x1</td>
<td>(C^*_1)</td>
</tr>
<tr>
<td>D11</td>
<td>(C^*_1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>y1</th>
<th>y2</th>
<th>y3</th>
<th>y4</th>
<th>y5</th>
<th>y6</th>
<th>y7</th>
</tr>
</thead>
</table>

...
### Optimal Alignment

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1$</td>
<td>$C_{11}^*$</td>
<td>$C_{21}^*$</td>
<td>$C_{31}^*$</td>
<td>$C_{41}^*$</td>
<td>$C_{51}^*$</td>
</tr>
<tr>
<td>$y_2$</td>
<td>$C_{12}^*$</td>
<td>$C_{22}^*$</td>
<td>$C_{32}^*$</td>
<td>$C_{42}^*$</td>
<td>$C_{52}^*$</td>
</tr>
<tr>
<td>$y_3$</td>
<td>$D_{11}$</td>
<td>$D_{21}$</td>
<td>$D_{31}$</td>
<td>$D_{41}$</td>
<td>$D_{51}$</td>
</tr>
<tr>
<td>$y_4$</td>
<td>$D_{12}$</td>
<td>$D_{22}$</td>
<td>$D_{32}$</td>
<td>$D_{42}$</td>
<td>$D_{52}$</td>
</tr>
<tr>
<td>$y_5$</td>
<td>$D_{13}$</td>
<td>$D_{23}$</td>
<td>$D_{33}$</td>
<td>$D_{43}$</td>
<td>$D_{53}$</td>
</tr>
<tr>
<td>$y_6$</td>
<td>$D_{14}$</td>
<td>$D_{24}$</td>
<td>$D_{34}$</td>
<td>$D_{44}$</td>
<td>$D_{54}$</td>
</tr>
<tr>
<td>$y_7$</td>
<td>$D_{15}$</td>
<td>$D_{25}$</td>
<td>$D_{35}$</td>
<td>$D_{45}$</td>
<td>$D_{55}$</td>
</tr>
</tbody>
</table>

...
### Optimal Alignment

<table>
<thead>
<tr>
<th></th>
<th>$D_{51}$</th>
<th>$D_{52}$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_5$</td>
<td>$C_{51}^*$</td>
<td>$C_{52}^*$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$D_{41}$</th>
<th>$D_{42}$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_4$</td>
<td>$C_{41}^*$</td>
<td>$C_{42}^*$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$D_{31}$</th>
<th>$D_{32}$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_3$</td>
<td>$C_{31}^*$</td>
<td>$C_{32}^*$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$D_{21}$</th>
<th>$D_{22}$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2$</td>
<td>$C_{21}^*$</td>
<td>$C_{22}^*$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$D_{11}$</th>
<th>$D_{12}$</th>
<th>$D_{13}$</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$C_{11}^*$</td>
<td>$C_{12}^*$</td>
<td>$C_{13}^*$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | $y_7$ |   |

...
Optimal Alignment

<table>
<thead>
<tr>
<th></th>
<th>$C^*_{51}$</th>
<th>$C^*_{52}$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_4$</td>
<td>$C^*_{41}$</td>
<td>$C^*_{42}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td>$C^*_{31}$</td>
<td>$C^*_{32}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td>$C^*_{21}$</td>
<td>$C^*_{22}$</td>
<td>$D_{23}$</td>
<td>$C^*_{23}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_1$</td>
<td>$C^*_{11}$</td>
<td>$C^*_{12}$</td>
<td>$D_{13}$</td>
<td>$C^*_{13}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$y_1$ $y_2$ $y_3$ $y_4$ $y_5$ $y_6$ $y_7$
Optimal Alignment

<table>
<thead>
<tr>
<th></th>
<th>$D_{51}$</th>
<th>$D_{52}$</th>
<th></th>
<th>$D_{33}$</th>
<th></th>
<th>$D_{13}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_5$</td>
<td>$C_{51}^*$</td>
<td>$C_{52}^*$</td>
<td></td>
<td>$C_{33}^*$</td>
<td></td>
<td>$C_{13}^*$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$C_{41}^*$</td>
<td>$C_{42}^*$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td>$C_{31}^*$</td>
<td>$C_{32}^*$</td>
<td>$D_{33}$</td>
<td>$C_{33}^*$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td>$C_{21}^*$</td>
<td>$C_{22}^*$</td>
<td>$D_{23}$</td>
<td>$C_{23}^*$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_1$</td>
<td>$C_{11}^*$</td>
<td>$C_{12}^*$</td>
<td>$D_{13}$</td>
<td>$C_{13}^*$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$y_1$, $y_2$, $y_3$, $y_4$, $y_5$, $y_6$, $y_7$
Optimal Alignment

until we recover the final value $C_{57}^*$
Optimal Alignment

Complexity: \(nm\) operations

... substantial improvement over \(\text{Delannoy}(n, m) \times \text{cost per path} \) ...
DTW distance

To recapitulate

• Sakoe & Chiba defined the distance

\[ d_{\text{DTW}}(x, y) = \min_{\pi \in \mathcal{A}(|x|, |y|)} \sum_{i=1} d(x_{\pi_1(i)}, y_{\pi_2(i)}) , \]

where \( d(x, y) \) is usually \( d(x, y) = \|x - y\| \).

• Can be computed in \( O(nm) \) iterations.

• Can be proved to be a distance (triangular inequality, etc...)

What are the strengths & weaknesses of the DTW?
Strengths of the DTW distance

- Intuitive, works well for simple examples in practice.

- Can be easily generalized to time-series in **metric spaces** - just need \( d(x_i, y_j) \)

- Used extensively in **information retrieval / nearest neighbour** search:
  - Given \( x \), scan in a large database and return its closest matches
  - Clever approaches to speed up these searches

Image taken from [http://www.eng.chula.ac.th/](http://www.eng.chula.ac.th/) (Chulalongkorn University)
Weaknesses of the DTW distance

- The distance DTW is **NOT** a **negative definite kernel**. The similarity

\[ k_{\text{DTW}}(x, y) = e^{-d_{\text{DTW}}(x, y)} , \]

is **NOT** positive-definite in general.

- You can use it with a SVM... but you have to **tweak** it or be lucky

- More worryingly, DTW is a very arbitrary choice:

  Given \( x \) and \( y \), DTW quantifies their similarity by looking at the set of all costs

  \( \{ C_{x,y}(\pi), \pi \in \mathcal{A}(|x|, |y|) \} \)

  but only considers its **minimum**!

- This leads to **unexpected** and **counter-intuitive** behavior in some cases:
Weaknesses of the DTW distance

Alignments from 1 to D(n,m)

Plot of all 1683 costs $C_{X,Y}(\tau)$ (ordered), when offset=−1

MIN $C_i$: 5

$\sin(\pi t)$

X and Y time series

$\min C_i$ as a function of the offset

DTW score
A different idea, more robust

- Rather than the minimum, consider the soft-minimum of $C_{x,y}$:

$$\text{soft-minimum}(C_{x,y}) = - \log \sum_{\pi \in A(|x|,|y|)} e^{-C_{x,y}(\pi)}$$

- Since we need a similarity, we consider $\exp(-\text{soft-minimum})$,

$$k_{GA} = \sum_{\pi \in A(|x|,|y|)} e^{-C_{x,y}(\pi)}$$

- First proposed here! J.P. Vert, H. Saigo & Prof. Akutsu in a 2004 paper
- Also considered on trees currently (joint work with K.Shin & T. Kuboyama)
- Let’s compare $k_{DTW} = e^{-DTW}$ and $k_{GA}$
\( e^{-\min C(\pi)} \) vs \( e^{-\text{soft-min} C(\pi)} = \sum e^{-C_i} \)
Minimal-cost alignment vs. all alignments

- **Soft-minimum** is intuitively more appealing than minimum.

Yet, not enough... two important issues remain:

- Do we have to sum over all $A(|x|, |y|)$ alignments to compute $k_{GA}$?
- $k_{DTW}$ is **NOT** positive definite, what about $k_{GA}$?

These two questions were answered in our ICASSP 2007 paper:
*A kernel for Time-Series based on Global Alignments*, M.C, J.-P. Vert, O. Birkenes, T. Matsui
All alignments: cheap to compute

- Do we have to sum over all $A(|x|, |y|)$ alignments to compute $k_{GA}$? NO
  - Computing $k_{GA}$ has the same complexity than DTW: $O(nm)$.
  - Change Bellman recursion $C^*_{i+1,j+1} = \min(C^*_{i+1,j}, C^*_{ij}, C^*_{i,j+1}) + D_{i+1,j+1}$

  \[
  x_{i+1} \quad K_{i+1,j} \quad K_{i+1,j+1} \\
  x_i \quad K_{ij} \quad K_{i,j+1} \\
  \quad y_j \quad y_{j+1}
  \]

  to $K_{i+1,j+1} = (K_{i+1,j} + K_{ij} + K_{i,j+1}) e^{-D_{i+1,j+1}}$

  - Recover kernel value as $k_{GA}(x,y) = K_{|x|,|y|}$.
  - Similar to the work of Vert-Saigo-Akutsu.
All alignments: Positive Definite

- $k_{\text{DTW}}$ is **NOT** positive definite, what about $k_{\text{GA}}$? **YES, BUT...**
  - $k_{\text{GA}}$ is positive definite if the function $f(x, y) \overset{\text{def}}{=} e^{-d(x,y)}$ is such that
    \[
    \frac{f}{1 + f}
    \]
    is a positive definite kernel.
  - Simple trick to define functions $f$: take a p.d. kernel $\kappa < 1$, define
    \[
    f \overset{\text{def}}{=} \frac{\kappa}{1 - \kappa}.
    \]
    - in such a case,
    \[
    \frac{f}{1 + f} = \frac{\frac{\kappa}{1 - \kappa}}{1 + \frac{\kappa}{1 - \kappa}} = \kappa
    \]
    which is positive definite.
  - Very different proof, quite involved... please check the paper.
Still... a few challenges

The global alignment kernel $k_{GA}$ is not without problems

- $k_{GA}$ can be diagonally dominant: $k_{GA}(x, x) \gg 1$ but $k_{GA}(x, y) \approx 0$.
- the condition $f/(1 + f)$ is positive definite is not well-understood.
- the quadratic $O(nm)$ complexity is still too high for large-scale applications.

In more recent work I look at these 3 different problems.

Cuturi, *Fast Global Alignment Kernels* (ICML 2011)
1. Diagonal Dominance

**Problem:** sometimes \( k_{GA}(x, x) \gg 1 \) but \( k_{GA}(x, y) \approx 0 \).

- **Solution:** use a **negative definite** distance \( d \) (\( \leftrightarrow \) **infinitely divisible** kernel \( \kappa \)), i.e. such that \( \kappa(x, y) \overset{\text{def}}{=} e^{-\lambda d(x, y)} \) is positive definite \( \forall \lambda > 0 \).

- When \( d \) is scaled by \( \lambda \to \infty \),

\[
k_{GA}(x, y) = \sum_{\pi \in \mathcal{A}(|x|, |y|)} e^{-\lambda C_{x,y}(\pi)} = 1_{\{x=y\}} \text{card } \mathcal{A}(|x|, |x|) = 1_{\{x=y\}} \text{Delannoy}(|x|)
\]

yet, when \( \lambda = 0 \),

\[
k_{GA}(x, y) = \sum_{\pi \in \mathcal{A}(|x|, |y|)} e^{0} = \text{card } \mathcal{A}(|x|, |y|) = \text{Delannoy}(|x|, |y|)
\]

- Given a database \( x_1, \ldots, x_N \), the Gram matrix varies between
  - \( \lambda = 0 \): the matrix \([\text{Delannoy}(|x_i|, |x_j|)]\)
  - \( \lambda \to \infty \): the Diagonal matrix \( \text{diag} (\text{Delannoy}(|x_i|)) \).
1. Diagonal Dominance

- if $|x_i| = |x_j|$, we can tune $\lambda$ to solve diagonal dominance.
- if $x_i \neq x_j$,
  - Can prove a bound on the spectrum of the Delannoy $D(n, m)$ matrix,
    \[ \sum_{i,j=1, i \neq j}^{n} D(i, j) > \left(1 - \frac{n}{9n-1}\right) \sum_{i=1}^{n} D_i. \]
  - $k_{GA}(x, y)$ with $\lambda = 0$ is significantly different from 0 if $\frac{1}{2} < \frac{|x|}{|y|} < 2$.

**Conclusion**: using a scaled n.d. distance $\lambda d$, diagonal dominance can be avoided when lengths are not too different.
2. New results: Geometric Divisibility

Definition 2 (Geometric Divisibility). Let \( f \) be a nonnegative valued function on \( \mathcal{X} \times \mathcal{X} \). \( f \) is said to be geometrically divisible (g.d.) if \( f/(1 + f) \) is positive definite.

Remark 1. If \( f \) is g.d. and \( \kappa \overset{\text{def}}{=} f/(1 + f) \) then \( f = \sum_{i=1}^{\infty} \kappa^i \) is necessarily p.d.

Lemma 2. The Gaussian kernel \( \kappa_\sigma \) is not geometrically divisible.

Lemma 3. For an infinitely divisible kernel \( \kappa \) such that \( 0 < \kappa < 1 \), \( \kappa/(1 - \kappa) \) is both geometrically divisible and infinitely divisible.

Motivated by these results, I propose to use the following distance in \( k_{GA} \),

\[
d(x, y) \overset{\text{def}}{=} \frac{1}{2\sigma^2} ||x - y||^2 + \log \left( 2 - e^{-\frac{||x - y||^2}{2\sigma^2}} \right).
\]
3. Speeding up $\kappa_{GA}$: old ideas from DTW

Itakura (75) and Sakoe-Chiba (78) propose to speed up the DTW computation by ignoring zones in the grid.
3. Speeding up $\kappa_{\text{GA}}$: old ideas from DTW

Decide a-priori that some paths are unlikely to be of interest.
3. Speeding up $k_{GA}$: old ideas from DTW

Easily done by setting distance $D_{ij} = \infty$ when $|i - j| > T$. 
3. Speeding up $k_{GA}$: old ideas from DTW

![Diagram showing distance matrix]

**Speed up: from $O(nm)$ to $O(T \min(n, m))$.**
3. Speeding up $k_{GA}$: old ideas from DTW

Yet, this can be suboptimal! Not guaranteed to find best path!
3. Speeding up $k_{GA}$

- In kernel methods, such weighting schemes need to preserve **positive definiteness**.

- Consider p.d. kernels $\omega(i, j)$ that only depend on $|i - j|$,

\[ \omega(i, j) = \psi(|i - j|), \]

where $\psi$ is a real-valued function on $\mathbb{N}$.

- Such kernels on integers are also known as **Toeplitz kernels**.

**Definition 3.** A Toeplitz kernel $\omega$ is compactly supported of order $T \in \mathbb{N}$ if for $q \geq T$, $\psi(q) = 0$ and $\psi(T - 1) \neq 0$. 

3. Speeding up $k_{GA}$

- Using such a kernel within GA kernels has obvious advantages

**Theorem 2.** Let $\kappa$ be a kernel on $\mathcal{X} \times \mathcal{X}$ and $\omega$ a compactly supported Toeplitz kernel of order $T$. Then using $\frac{\omega \kappa}{1 - \omega \kappa}$ as a local kernel, $k_{GA}(x, y)$ can be computed with $O(T \min(n, m))$ operations. Furthermore, $k_{GA}(x, y)$ is null when $|n - m| > T$.

- Example: Triangular Kernel

\[
\omega(i, j) = \left(1 - \frac{|i - j|}{T}\right)_+. 
\]
3. Speeding up $k_{GA}$

Using a triangular kernel $\omega$ and a kernel $\kappa$, $k_{GA}$ is also sped up to $O(T \min(n, m))$...

Here $K_{i,j}$ stands for $(\cdot \cdot_{1-}) (\omega \otimes \kappa ((i, x_i), (j, y_j)))$
Experimental Results: Classifying Time Series

Benchmark Datasets (UCI repository) + **PEMS database** which we assembled

<table>
<thead>
<tr>
<th>Database</th>
<th>$d$</th>
<th>$n, \text{med}(n)$</th>
<th>classes</th>
<th># points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese Vowels</td>
<td>12</td>
<td>7-29, 15</td>
<td>9</td>
<td>640</td>
</tr>
<tr>
<td>Libras</td>
<td>2</td>
<td>45</td>
<td>15</td>
<td>945</td>
</tr>
<tr>
<td>Handwritten Characters</td>
<td>3</td>
<td>60-182, 122</td>
<td>20</td>
<td>2858</td>
</tr>
<tr>
<td>AUSLAN</td>
<td>22</td>
<td>45-136, 55</td>
<td>95</td>
<td>2465</td>
</tr>
<tr>
<td>PEMS</td>
<td>963</td>
<td>144</td>
<td>7</td>
<td>440</td>
</tr>
</tbody>
</table>

We consider the DTW kernel $k_{\text{DTW}}$ and a few more...

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Parameters</th>
<th>Parameter Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{\text{DTW}}$</td>
<td>$t$</td>
<td>$t \in {0.2, 0.5, 1, 2, 5} \cdot \text{med}(\text{DTW}(x, x))$</td>
</tr>
<tr>
<td>$k_{SC}$</td>
<td>$t, T$</td>
<td>$t \in {0.2, 0.5, 1, 2, 5} \cdot \text{med}(\text{DTW}_{\text{SC}}(x, y)), \ T \in {0.25, 0.5} \cdot \text{med}(</td>
</tr>
<tr>
<td>$k_{\text{DTAK}}$</td>
<td>$t, \sigma$</td>
<td>$t \in {0.2, 0.5, 1, 2, 5} \cdot \text{med}(− \log k_{\text{DTAK}}(x, y)), \ \sigma \in {0.2, 0.5, 1, 2} \cdot \text{med}(</td>
</tr>
<tr>
<td>$k_{\text{GA}}$</td>
<td>$\sigma$</td>
<td>$\sigma \in {0.2, 0.5, 1, 2, 5} \cdot \text{med}(</td>
</tr>
<tr>
<td>$k_{\text{TGA}}$</td>
<td>$\sigma, T$</td>
<td>$\sigma \in {0.2, 0.5, 1, 2, 5} \cdot \text{med}(</td>
</tr>
</tbody>
</table>
Results averaged on 3-fold 3-repeats cross validations.
Parameters selected within training folds using 3-fold 2-repeats.
### Experimental Results

<table>
<thead>
<tr>
<th>Language</th>
<th>Mean/std of error rate (%)</th>
<th>Mean/std of time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUSLAN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LIBRAS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEMS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Comparing the effect of $T$ (as fraction of median length) on speed and classification performance.
Better not use DTW with a kernel machine (e.g. SVM's), try $k_{GA}$ instead