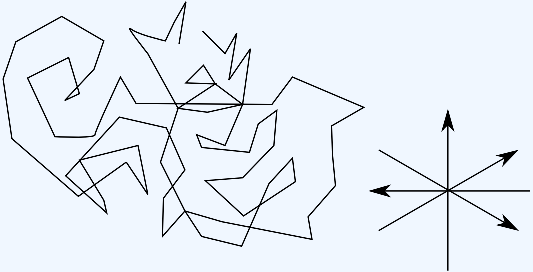
Extending a Novel Approach for Clustering Time-Series

Treating Multivariate Time-Series as Polygonal Curves

Stefan Meintrup
Alexander Munteanu
Dennis Rohde

TU Dortmund University

October 1, 2019



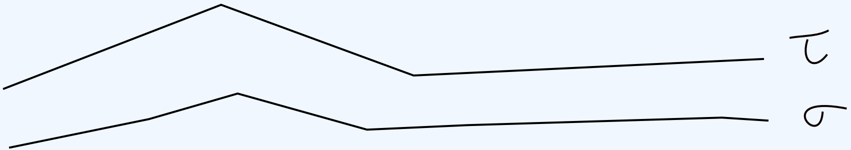
Motivation

Fréchet distance is **building block** in many (machine learning) applications

- morphing
- protein structure alignment
- handwriting recognition
- **clustering of time-series** (weather, large physical experiments, stock...)
  - ☞ compensate different sampling-rates and inhomogeneous lengths by only comparing the "shape"
    - Driemel et al. (SODA 2015): Clustering time-series under the Fréchet distance (univariate)
    - (NeurIPS 2019) this work (multivariate)

- sum- or average-based distance? **intractable** for high dimensions (see Efrat (2007): Curve Matching, Time Warping ...)
- bottle-neck distance: **Fréchet distance** a.k.a. "dog-man distance"

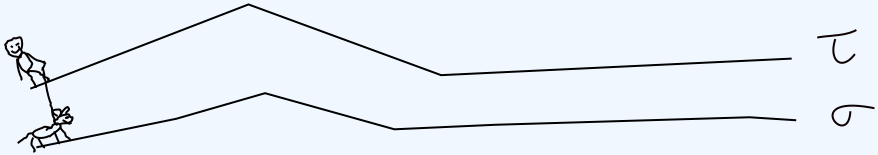
$$d_F : (\sigma, \tau) \mapsto \inf_{h\,:\,[0,1]\to[0,1]\text{ reparam.}} \max_{t\in[0,1]} \|\sigma(t) - \tau(h(t))\|_2$$



How to Compare Polygonal Curves?

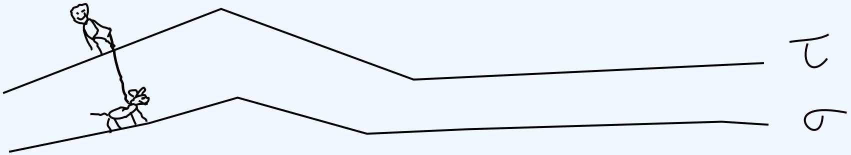
- sum- or average-based distance? **intractable** for high dimensions ↯ (see Efrat (2007): Curve Matching, Time Warping ...)
- bottle-neck distance: **Fréchet distance** a.k.a. "dog-man distance"

$$d_F : (\sigma, \tau) \mapsto \inf_{h:\ [0,1]\to[0,1]\ \text{reparam.}} \max_{t\in[0,1]} \|\sigma(t) - \tau(h(t))\|_2$$



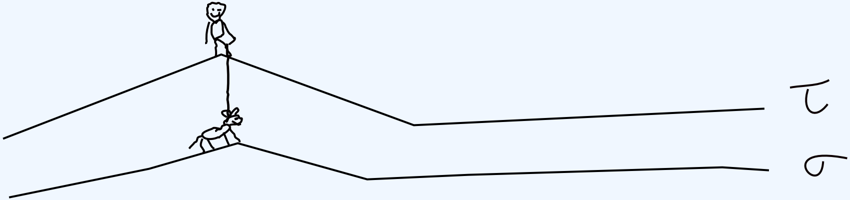
- sum- or average-based distance? **intractable** for high dimensions ↯ (see Efrat (2007): Curve Matching, Time Warping...)
- bottle-neck distance: **Fréchet distance** a.k.a. "dog-man distance"

$$d_F : (\sigma, \tau) \mapsto \inf_{h : [0,1] \to [0,1] \text{ reparam.}} \max_{t \in [0,1]} \|\sigma(t) - \tau(h(t))\|_2$$



- sum- or average-based distance? **intractable** for high dimensions ↯ (see Efrat (2007): Curve Matching, Time Warping...)
- bottle-neck distance: **Fréchet distance** a.k.a. "dog-man distance"

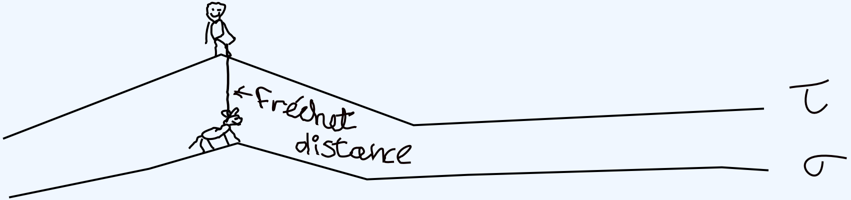
$$d_F : (\sigma, \tau) \mapsto \inf_{h:\ [0,1]\to[0,1]\ \text{reparam.}} \max_{t\in[0,1]} \|\sigma(t) - \tau(h(t))\|_2$$



How to Compare Polygonal Curves?

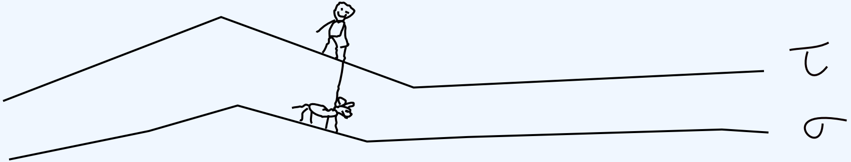
- sum- or average-based distance? **intractable** for high dimensions ↯ (see Efrat (2007): Curve Matching, Time Warping …)
- bottle-neck distance: **Fréchet distance** a.k.a. “dog-man distance”

$$d_F : (\sigma, \tau) \mapsto \inf_{h:\ [0,1]\to[0,1] \text{ reparam.}} \ \max_{t\in[0,1]} \|\sigma(t) - \tau(h(t))\|_2$$



- sum- or average-based distance? **intractable** for high dimensions (see Efrat (2007): Curve Matching, Time Warping..)
- bottle-neck distance: **Fréchet distance** a.k.a. "dog-man distance"

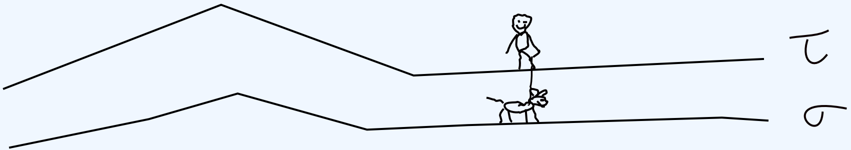
$$d_F : (\sigma, \tau) \mapsto \inf_{h:\ [0,1]\to[0,1]\ \text{reparam.}} \max_{t\in[0,1]} \|\sigma(t) - \tau(h(t))\|_2$$



How to Compare Polygonal Curves?

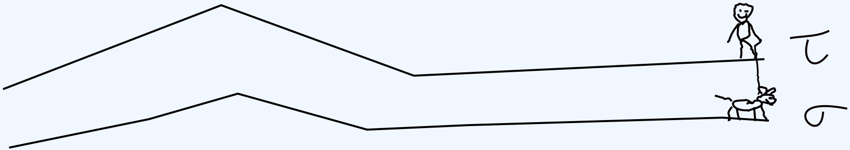
- sum- or average-based distance? **intractable** for high dimensions (see Efrat (2007): Curve Matching, Time Warping...)
- bottle-neck distance: **Fréchet distance** a.k.a. "dog-man distance"

$$d_F : (\sigma, \tau) \mapsto \inf_{h:\,[0,1]\to[0,1] \text{ reparam.}} \max_{t\in[0,1]} \|\sigma(t) - \tau(h(t))\|_2$$

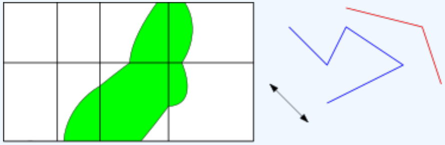


- sum- or average-based distance? **intractable** for high dimensions ↯ (see Efrat (2007): Curve Matching, Time Warping ...)
- bottle-neck distance: **Fréchet distance** a.k.a. "dog-man distance"

$$d_F : (\sigma, \tau) \mapsto \inf_{h:\,[0,1]\to[0,1]\text{ reparam.}} \max_{t\in[0,1]} \|\sigma(t) - \tau(h(t))\|_2$$



- **Alt-Godau Algorithm** (1995: Computing the Fréchet distance between two Polygonal Curves)
  - Running-time $\mathcal{O}(d \cdot m^2 \log(m))$
- There is no algorithm with running-time $\mathcal{O}(m^{2-\delta})$, for any $\delta > 0$, unless SETH' fails (Bringmann 2014: Why walking the dog takes time...)



What Happens in the Big Data Regime?

- n: (large) number of curves to cluster
- high complexity, say $m \in \Omega(n)$
- high-dimensional, say $d \in \Omega(n)$

$\rightarrow$ Running-time super-cubic in n in the worst case
  - **But** quadratic running-time is already considered intractable on Big Data

- Can we improve?

What Happens in the Big Data Regime?

- n: (large) number of curves to cluster
- high complexity, say $m \in \Omega(n)$ ☞ **parallelization (CUDA)**
- high-dimensional, say $d \in \Omega(n)$

$\rightarrow$ Running-time super-cubic in n in the worst case

- **But** quadratic running-time is already considered intractable on Big Data

- Can we improve?

What Happens in the Big Data Regime?

- n: (large) number of curves to cluster
- high complexity, say $m \in \Omega(n)$ ☞ **parallelization (CUDA)**
- high-dimensional, say $d \in \Omega(n)$ ☞ **dimension reduction (quality guarantee)**

$\rightarrow$ Running-time super-cubic in n in the worst case
  - **But** quadratic running-time is already considered intractable on Big Data

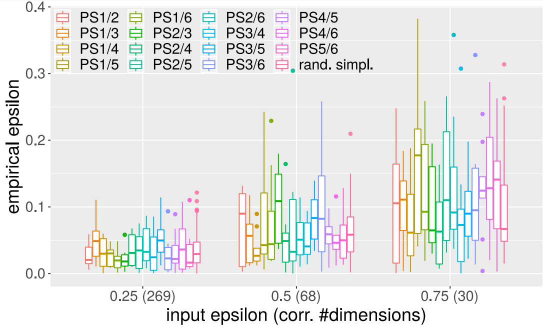
- Can we improve?

What Happens in the Big Data Regime?

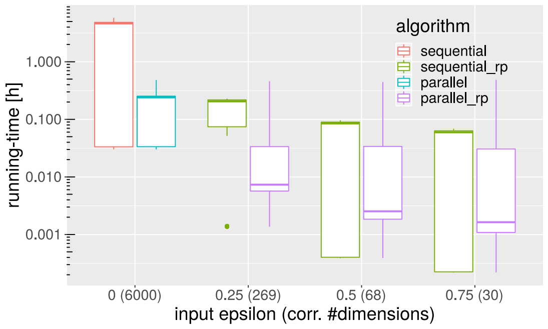
- n: (large) number of curves to cluster **subsampling (quality guarantee)**
- high complexity, say $m \in \Omega(n)$ ☞ **parallelization (CUDA)**
- high-dimensional, say $d \in \Omega(n)$ ☞ **dimension reduction (quality guarantee)**

→ Running-time super-cubic in n in the worst case
  - **But** quadratic running-time is already considered intractable on Big Data

- Can we improve?



(a) Quality



(b) Running-time