Learning dynamics with dynamical distances

From diffusion maps to commute maps and coherence

Ralf Banisch, with Cecilia Clementi, Frank Noé, and Péter Koltai
IPAM December 5th 2016

papers:

RB and Péter Koltai. Understanding the geometry of transport: diffusion maps for Lagrangian trajectory data unravel coherent sets. arXiv:1603.04709 (accepted in CHAOS)
Motivation: Time series

Brownian particle in a potential landscape

earthquake events in Southern California
Outline of the talk

• learning the geometry of point clouds: diffusion maps
• dynamical distances I: commute maps
• dynamical distances II: Time-averaged diffusion maps
Outline of the talk

- **learning the geometry of point clouds: diffusion maps**
- dynamical distances I: commute maps
- dynamical distances II: Time-averaged diffusion maps
Learning geometry: local vs. global

The game: Given data points in $\mathbb{R}^n$ that lie on an unknown submanifold $M$ and are distributed according to an unknown distribution $q$, learn the geometry of $M$!

When we play this game, we can trust the Euclidean distance locally, but not globally!
Learning geometry: local vs. global

**Diffusion maps idea:** Build a Markov chain on the data points. Base jump probabilities on Euclidean distance, but only allow local jumps. Then compute distances based on how the Markov chain traverses the dataset.

[Coifman & Lafon]
Learning geometry: constructing the Markov chain

Our data: $m$ data points $\{x_i\}_{i=1}^{m} \in \mathbb{R}^n$.

build a similarity matrix based on kernel function $h$

$$k_{\varepsilon}(x_i, x_j) = h\left(\frac{\|x_i - x_j\|^2}{\varepsilon}\right)$$

normalize

row-stochastic matrix

$$P_{\varepsilon, \alpha}(i, j) := \frac{k_{\varepsilon}^{(\alpha)}(x_i, x_j)}{d_{\varepsilon}^{(\alpha)}(x_i)}$$

Parameters:
- $\varepsilon > 0$ scaling parameter
- $\alpha \in [0, 1]$
Learning geometry: from points to manifolds

**Central result** in diffusion maps: In the limit of infinite data and for $\varepsilon \to 0$, the generator of the Markov chain converges to a differential operator on $\mathcal{M}$. [Coifman & Lafon 2006]

\[
\lim_{\varepsilon \to 0} \lim_{m \to \infty} L_{\varepsilon, \alpha} f = \Delta f + (2 - 2\alpha) \nabla f \cdot \frac{\nabla q}{q}
\]

\[
L_{\varepsilon, \alpha} = \varepsilon^{-1} (P_{\varepsilon, \alpha} - I)
\]
Learning geometry: from points to manifolds

**Central result** in diffusion maps: In the limit of infinite data and for $\varepsilon \to 0$, the generator of the Markov chain converges to a differential operator on $M$. [Coifman & Lafon 2006]

$$\lim_{\varepsilon \to 0} \lim_{m \to \infty} L_{\varepsilon, \alpha} f = \Delta f + (2 - 2\alpha) \nabla f \cdot \frac{\nabla q}{q}$$

**L_{\varepsilon, \alpha} = \varepsilon^{-1} (P_{\varepsilon, \alpha} - I)**

**Why is this nice?** Dominant eigenfunctions of $\Delta$ are **good coordinates** on $M$, and we can approximate the eigenfunctions of $\Delta$ by eigenfunctions of $L_{\varepsilon, 1}$.

2nd eigenfunction of $L_{\varepsilon, 1}$  
3rd eigenfunction of $L_{\varepsilon, 1}$  
embedding
Diffusion distance

\[ D^2_{\tau}(x_1, x_2) = \|p_\tau(\cdot|x_1) - p_\tau(\cdot|x_2)\|_{\pi^{-1}}^2 = \sum_{j=1}^{m} \lambda_j^{2\tau} (\psi_j(x_1) - \psi_j(x_2))^2 \]

probability distribution after \(\tau\) steps of diffusion on the graph with start at \(x_1\)
Outline of the talk

• learning the geometry of point clouds: diffusion maps

• dynamical distances I: commute maps

• dynamical distances II: Time-averaged diffusion maps
we can describe the evolution of the dynamics of $X_t$ via a family of transition functions:

$$p_t(x, A) = \mathbb{P}[X_t \in A | X_0 = x]$$

- **invariant** measure $\pi$:
  $$\int p_t(x, A) \pi(dx) = \pi(A)$$

- the transition functions induce a one-parameter family of **propagators**:

$$\rho_{t+\tau}(y) = \mathcal{P}_\tau \rho_t(y) = \int \rho_t(x) p_\tau(x, y) dx$$
Technical slide: Transfer operators

\[ \rho_{t+\tau}(y) = \mathcal{P}_\tau \rho_t(y) = \int \rho_t(x)p_\tau(x, y) \, dx \]

- in \( L^2(\pi) \) and under appropriate conditions, the propagators have a spectral decomposition with **dominant** isolated real eigenvalues:

\[ 1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \]
Technical slide: Transfer operators

\[ \rho_{t+\tau}(y) = \mathcal{P}_\tau \rho_t(y) = \int \rho_t(x) p_\tau(x, y) \, dx \]

• in \( L^2(\pi) \) and under appropriate conditions, the propagators have a spectral decomposition with dominant isolated real eigenvalues:

\[ 1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \]

• **reduced kinetic model:** Markov State Model (MSM) associated with partition of state space

\[
\begin{align*}
P_\tau & = Q^T \mathcal{P}_\tau Q \\
P_\tau(i, j) & = \mathbf{P} \left[ X_{t+\tau} \in \chi_j \mid X_t \in \chi_i \right]
\end{align*}
\]

variational principle: \( \hat{\lambda}_i(\tau) \leq \lambda_i(\tau) \)

[Deuflhard, Huisinga, Fischer, Schütte 2000]
Kinetic distance

- the transition functions induce a one-parameter family of propagators:

\[ \rho_{t+\tau}(y) = \mathcal{P}_\tau \rho_t(y) = \int \rho_t(x) p_{\tau}(x, y) dx \]

- \( \tau \)-kinetic distance

\[ D^2_\tau(x_1, x_2) = \| p_\tau(\cdot|x_1) - p_\tau(\cdot|x_2) \|_{\pi-1}^2 = \sum_{j=1}^{m} \lambda_j^2(\tau) (\psi_j(x_1) - \psi_j(x_2))^2 \]

probability distribution after dynamics is evolved for \( \tau \) steps with start at \( x_1 \)
Kinetic distance: Example

results are strongly $\tau$ dependent!
Kinetic distance: Example

\[ D_\tau(x_1, x_2) \]

\[ k\text{-means based on } D_\tau(x_1, x_2) \]

results are strongly \( \tau \) dependent!

[Noé and Clementi 2015]
Commuting distance

- **Dynamical** diffusion distance

\[ D^2_{\tau}(x_1, x_2) = \| p_\tau(\cdot | x_1) - p_\tau(\cdot | x_2) \|_{2, \tau}^2 = \sum_{j=1}^{m} \lambda_j^2(\tau) (\psi_j(x_1) - \psi_j(x_2))^2 \]

- To get rid of dependence on lagtime \( \tau \), introduce **commute distance**:

\[ d^2_{\text{comm}}(x_1, x_2) = \int_0^\infty D^2_{\tau}(x_1, x_2) d\tau = \frac{1}{2} \sum_{j=2}^{m} t_j (\psi_j(x_1) - \psi_j(x_2))^2 \]

- **Kinetic content** explained by the first \( m' \) eigenvalues:

\[ c_{m'} = \frac{\sum_{j=2}^{m'} t_j}{K}, \quad K = \sum_{j=2}^{m} t_j \]

[Noé, RB and Clementi, JCTC 2016]
Commute distance

[Noé, RB and Clementi, JCTC 2016]

τ-kinetic distance

Cumulative Kinetic Content

commute distance

Cumulative Kinetic Content

kinetic content explained

k-means
Commute distance

\[ \text{kinetic model quality} \]

\[ \text{Commute distance}[\text{Noé, RB and Clementi, JCTC 2016}] \]
Commute distance vs. commute time

[Noé, RB and Clementi, JCTC 2016]

- In discrete state spaces, the (full) commute distance equals the commute time:
  \[
d^2_{\text{comm}}(x_i, x_j) = \frac{t_{ij} + t_{ji}}{2}
\]

- If we only use the largest \( m' \) eigenvalues, we get an upper bound:
  \[
  \left[ d^{(m')}_{\text{comm}}(x_i, x_j) \right]^2 \leq \frac{t_{ij} + t_{ji}}{2}
  \]
Estimation from data

\[ d_{\text{comm}}^2(x_i, x_j) = \frac{1}{2} \sum_{j=2}^{m} t_j (\psi(x_i) - \psi(x_j))^2, \quad \lambda_j(\tau) = \exp(-t_j/\tau) \]

VAC / EDMD:

needs estimation

[Noé, Nüske 2013]
[Williams, Kevrekidis, Rowley 2014]
Estimation from data

\[ d_{\text{comm}}^2(x_i, x_j) = \frac{1}{2} \sum_{j=2}^{m} t_j (\psi(x_i) - \psi(x_j))^2, \quad \lambda_j(\tau) = \exp(-t_j/\tau) \]

VAC / EDMD:

- given data in form of simulation trajectories \( x_{t,i} \) (\( t \) = time, \( i \) = coordinate), form the two matrices

\[
X_0 = \begin{bmatrix}
x_{0,1} & \cdots & x_{0,n} \\
\vdots & \ddots & \vdots \\
x_{T-\tau,1} & \cdots & x_{T-\tau,n}
\end{bmatrix} \quad X_\tau = \begin{bmatrix}
x_{0,\tau} & \cdots & x_{\tau,n} \\
\vdots & \ddots & \vdots \\
x_{T,1} & \cdots & x_{T,n}
\end{bmatrix}
\]

- compute the moment matrices \( C_{00} = X_0^T X_0, \quad C_{0\tau} = X_0^T X_\tau \)

- then solve generalised eigenvalue problem: \( C_{0\tau} R = C_{00} R A \)

- the eigenvectors are approximated as \( \Psi \approx X_0 R \)

- the commute map is given as \( \Psi' = X_0 R \text{diag} \left( \sqrt{\hat{t}_i/2} \right), \) with \( \hat{t}_i = -\tau / \ln \Lambda_{ii}(\tau) \)
Example: BPTI

- simulation data
- VAC/EDMD
- approximate eigenfunctions
- embedding: commute map, τ-kinetic map, …
- k-means
- coarse-grained kinetic model (MSM)

Data: 1 ms simulation of BPTI produced by Anton supercomputer

Variational principle: \( \hat{t}_i \leq t_i \)
Summary of Part II

- given data in form of simulation trajectories $x_{t,i}$, we can use VAC/EDMD to approximate the dominant time scales and eigenfunctions of the propagator.

- from these eigenfunctions, we can compute the commute map embedding.

$$\Psi' = X_0 R \text{diag} \left( \sqrt{\hat{t}_i/2} \right)$$

- the commute distance is the Euclidean distance in the commute map space.

$$d_{\text{comm}}^2(x_i, x_j) = \frac{1}{2} \sum_{j=2}^{m} t_j (\psi(x_j) - \psi(x_j))^2 = ||\psi'_i - \psi'_j||^2$$

- with geometric clustering techniques such as k-means, we can construct coarse grained kinetic models (MSMs) from the commute map.

- the resulting models outperform models constructed from different embeddings.
Outline of the talk

- learning the geometry of point clouds: diffusion maps
- dynamical distances I: commute maps
- dynamical distances II: Time-averaged diffusion maps
Motivation: What are coherent sets?
The game: Given \(mT\) data points which are organised as \(m\) trajectories produced from an unknown dynamical system \(\Phi_t\) sampled at \(T\) time points \(I_t = \{t_0, \ldots, t_{T-1}\}\), i.e. given the dataset

\[
X = \{x_t^i := \Phi_t x^i : i = 1, \ldots, m; \ t \in I_t\},
\]

learn something about the dynamics of \(\Phi_t\)!

Idea: Build a Markov chain on the trajectory data! Use diffusion maps to jump between points in the same time slice.
Time-averaged diffusion maps

The game: Given $mT$ data points which are organised as $m$ trajectories produced from an unknown dynamical system $\Phi_t$ sampled at $T$ time points $I_t = \{t_0, \ldots, t_{T-1}\}$, i.e. given the dataset

$$X = \{x^i_t := \Phi_t x^i : i = 1, \ldots, m; \ t \in I_t\},$$

learn something about the dynamics of $\Phi_t$!

Idea: Build a Markov chain on the trajectory data! Use diffusion maps to jump between points in the same time slice.

Average over time slices, $\alpha = 1/2$:

$$Q_{\varepsilon} = \frac{1}{T} \sum_{t \in I_t} P_{\varepsilon, t}$$

average over time slices diffusion maps at time slice $t$
Time-averaged diffusion maps

Central result: If $\Phi_t$ is a diffeomorphism (i.e. the flow map of an ODE), then the limit of infinite data and for small $\varepsilon$, $Q_\varepsilon$ converges to an operator (as in diffusion maps):

$$\lim_{\varepsilon \to 0} \lim_{m \to \infty} \frac{1}{\varepsilon} (Q_\varepsilon - \text{Id}) = \frac{1}{2T} \sum_{t \in I_t} p_t^* \Delta q_t p_t$$

$$\Delta q f = q^{-1} \nabla \cdot (q \nabla f)$$

compare with dynamic Laplacian:

$$\hat{\Delta} = \frac{1}{2} (\Delta + p_t^* \Delta p_t)$$

The dominant eigenfunctions $\{\Psi_1, \ldots, \Psi_d\}$ of $Q_\varepsilon$ can be used to define a time-averaged diffusion distance:

$$d^2_T(x_1, x_2) = \sum_{j=1}^{m} \lambda_j^2 (\psi_j(x_1) - \psi_j(x_2))^2$$
Some related work

**Coherent Sets:**

- Shape coherence: T. Ma and E. Bollt, 2014

**Clustering in trajectory space:**

- Dynamical distance + spectral clustering: A. Hadjighasem et. al., 2015

**Transfer operator approximation:**

- DMD and EDMD: I. Mezic and others, 2013-2016
- Radial basis function collocation: O. Junge and A. Denner, 2016
Double gyre: Eigenvalues

We consider a data-rich case first: 20,000 trajectories with initial conditions on uniform grid.

eigenvalues of \( L_\varepsilon = \varepsilon^{-1} (Q_\varepsilon - I) \)
Double gyre: 3-Clustering & Eigenfunctions

We consider a data-rich case first: 20,000 trajectories with initial conditions on uniform grid.

\[
\varepsilon = 0.004
\]

Eigenvalues of \( L_\varepsilon = \varepsilon^{-1} (Q_\varepsilon - I) \)
We randomly select 500 of the 20,000 trajectories, and for those we destroy 80% of the entries. 97.5% of the data is destroyed!
Quasiperiodic Bickley jet: Eigenvalues  

12,000 trajectories, flow time \( t = 40 \) days.

9 clusters

eigenvalues of \( L_\varepsilon = \varepsilon^{-1} (Q_\varepsilon - I) \)

[RB & Koltaï 2016]

ralfbanisch.shinyapps.io/shiny_bickley/
Ocean drifter data

Data: 2,267 drifters in the global ocean, with monthly positions from Jan 2005 - Dec 2009.

This data is sparse: On average only 38% of all trajectories available at any time instant.
Ocean drifter data

[RB & Kolta 2016, Froyland & Padberg-Gehle 2015]

embedding

5-clustering

FIG. 19: 5-clustering of drifter data ($K = 5$ in order to attempt a maximum membership value of less than 1, corresponding to a) drifter positions January 2005, (b) drifter positions July 2007, (c) drifter positions December 2009. Animation of the cluster motion 2005–2009 in online version (Multimedia view).

We remark that while the results in Ref. 19 are of a...
ABC flow [RB & Koltai 2016]

64,000 trajectories, flow time $t=20$.

embedding
Change point detection

sliding window analysis:
Summary

• it is possible to define **dynamical distances** which capture essential dynamical features of interest: **slow dynamical modes, coherence,**...

• there is always a connection to an analytical transfer operator framework in terms of dominant eigenfunctions and eigenvalues.

• direct numerical estimation from trajectory data possible.

• the dynamical distances define a mapping to a low-dimensional space in which geometry-based machine learning methods can be applied.

• the resulting reduced-order kinetic models have superior accuracy.
Summary

• it is possible to define dynamical distances which capture essential dynamical features of interest: slow dynamical modes, coherence,…

• there is always a connection to an analytical transfer operator framework in terms of dominant eigenfunctions and eigenvalues.

• direct numerical estimation from trajectory data possible.

• the dynamical distances define a mapping to a low-dimensional space in which geometry-based machine learning methods can be applied.

• the resulting reduced-order kinetic models have superior accuracy.

useful links:

• papers: arxiv.org/abs/1603.04709 and DOI:10.1021/acs.jctc.6b00762

• app: ralfbanisch.shinyapps.io/shiny_bickley/
  and https://github.com/ralfbanisch/shiny-diffusion-maps

• PyEMMA analysis toolkit: http://pyemma.org