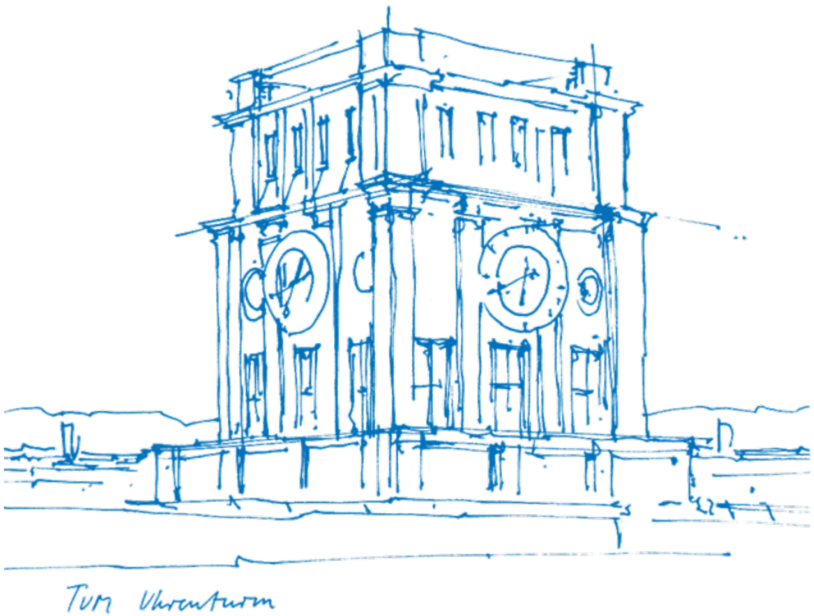
Extracting dynamical systems from data

Felix Dietrich



Outline

Extracting dynamical systems from data

1. Approximating functions from data
2. Vector fields from observations
3. Time-delay embedding

Approximating functions from data

Prototypical problem: supervised learning

The prototypical problem of machine learning is called “supervised learning”: Given some data $X = \left\{x^{(k)}\right\}_{k=1}^{N} \subset \mathbb{R}^n$, and function values $F = \left\{f\left(x^{(k)}\right)\right\}_{k=1}^{N} \subset \mathbb{R}^d$, construct a function $\hat{f} : \mathbb{R}^n \to \mathbb{R}^d$ such that the following error is minimal:

$$e\left(\hat{f}\right) = \|f(X) - \hat{f}(X)\|^2 = \|F - \hat{f}(X)\|^2. \tag{1}$$

Challenge: how can we minimize the error e over “all” functions $\hat{f}$?

Approximating functions from data

Prototypical problem: supervised learning

The prototypical problem of machine learning is called “supervised learning”:
Given some data $X = \left\{x^{(k)}\right\}_{k=1}^{N} \subset \mathbb{R}^n$, and function values $F = \left\{f\left(x^{(k)}\right)\right\}_{k=1}^{N} \subset \mathbb{R}^d$, construct a function $\hat{f} : \mathbb{R}^n \to \mathbb{R}^d$ such that the following error is minimal:

$$e\left(\hat{f}\right) = \|f(X) - \hat{f}(X)\|^2 = \|F - \hat{f}(X)\|^2. \tag{1}$$

Challenge: how can we minimize the error e over “all” functions $\hat{f}$?
(Partial) answer: pick a particular subset of functions and search there.

Approximating functions from data

Starting simple - linear functions

A linear function between two Euclidean spaces $\mathbb{R}^n, \mathbb{R}^d$ with $n, d \in \mathbb{N}$ is a map $f_{\text{linear}} : \mathbb{R}^n \to \mathbb{R}^d$, such that for $x \in \mathbb{R}^n$,

$$f_{\text{linear}}(x) = Ax + b \in \mathbb{R}^d \tag{2}$$

for some matrix $A \in \mathbb{R}^{d \times n}$ and $b \in \mathbb{R}^d$. [Note: we ignore affine maps and set $b = 0$.]

Approximating functions from data

Starting simple - linear functions

A linear function between two Euclidean spaces $\mathbb{R}^n, \mathbb{R}^d$ with $n, d \in \mathbb{N}$ is a map $f_{\text{linear}} : \mathbb{R}^n \to \mathbb{R}^d$, such that for $x \in \mathbb{R}^n$,

$$f_{\text{linear}}(x) = Ax + b \in \mathbb{R}^d \tag{2}$$

for some matrix $A \in \mathbb{R}^{d \times n}$ and $b \in \mathbb{R}^d$. [Note: we ignore affine maps and set $b = 0$.]

Approximation: least-squares minimization

How do we solve a supervised learning problem with a linear function? Assume we decided that our approximation function should be linear. Then, minimizing the squared error leads to

$$\min_{\hat{f}} e\left(\hat{f}\right) = \min_{\hat{f}} \|F - \hat{f}(X)\|^2 = \min_{A} \|F - XA^T\|^2, \tag{3}$$

where $f(X) =: F \in \mathbb{R}^{N \times d}$, $X \in \mathbb{R}^{N \times n}$, and $A \in \mathbb{R}^{d \times n}$.

Approximating functions from data

Starting simple - linear functions

A linear function between two Euclidean spaces $\mathbb{R}^n, \mathbb{R}^d$ with $n, d \in \mathbb{N}$ is a map $f_{\text{linear}} : \mathbb{R}^n \to \mathbb{R}^d$, such that for $x \in \mathbb{R}^n$,

$$f_{\text{linear}}(x) = Ax + b \in \mathbb{R}^d \tag{2}$$

for some matrix $A \in \mathbb{R}^{d\times n}$ and $b \in \mathbb{R}^d$. [Note: we ignore affine maps and set $b = 0$.]

Approximation: least-squares minimization

How do we solve a supervised learning problem with a linear function? Assume we decided that our approximation function should be linear. Then, minimizing the squared error leads to

$$\min_{\hat{f}} e\left(\hat{f}\right) = \min_{\hat{f}} \|F - \hat{f}(X)\|^2 = \min_{A} \|F - XA^T\|^2, \tag{3}$$

where $f(X) =: F \in \mathbb{R}^{N\times d}$, $X \in \mathbb{R}^{N\times n}$, and $A \in \mathbb{R}^{d\times n}$.
Here, another important algorithm enters: *least-squares minimization*. Problem (3) has a closed form solution minimizing the least-squares error:

$$\hat{A}^T = (X^T X)^{-1} X^T F. \tag{4}$$

Approximating functions from data

Nonlinear functions

A particular representation of a nonlinear function that is used in many numerical algorithms: linear decomposition into nonlinear basis functions. Basic idea: write the unknown function f such that

$$f(x) = \sum_{l=1}^{L} c_l \phi_l(x) = C\phi(x),\ c_l \in \mathbb{R}^d, C = [c_1, \dots, c_L],\ \phi(x) = (\phi_1(x), \dots, \phi_L(x))^T. \tag{5}$$

If L is finite, a finite-dimensional space of functions f can be represented in this way. You may have seen this decomposition in

1. Fourier analysis, $\phi_l(x) = \exp(2\pi i l\ x)$
2. Taylor decomposition, $\phi_l(x) = (x_0 - x)^l$ or
3. in lecture four on neural networks, $\phi_l(x) = \tanh(W_l x + b_l)$.

Approximating functions from data

Nonlinear functions

A particular representation of a nonlinear function that is used in many numerical algorithms: linear decomposition into nonlinear basis functions. Basic idea: write the unknown function f such that

$$f(x) = \sum_{l=1}^{L} c_l \phi_l(x) = C\phi(x),\ c_l \in \mathbb{R}^d, C = [c_1, \dots, c_L],\ \phi(x) = (\phi_1(x), \dots, \phi_L(x))^T. \tag{5}$$

If L is finite, a finite-dimensional space of functions f can be represented in this way. You may have seen this decomposition in

1. Fourier analysis, $\phi_l(x) = \exp(2\pi il\ x)$
2. Taylor decomposition, $\phi_l(x) = (x_0 - x)^l$ or
3. in lecture four on neural networks, $\phi_l(x) = \mathtt{tanh}(W_l x + b_l)$.

Benefit of linear decomposition

We can use least-squares minimization again! But be careful: the solution is not unique if the ϕ_l are not orthogonal (that means the inner product $\langle \phi_l, \phi_k \rangle$ may not be zero).

Approximating functions from data

Nonlinear functions

Consider special functions ϕ, so-called *radial basis functions* [Schölkopf and Smola, 2018]:

$$\phi_l(x) = h(r) = \exp\left(-r^2/\varepsilon^2\right),\ r := \|x_l - x\|. \tag{6}$$

The point x_l is the center of the basis function (where it attains its maximum value 1), and the parameter ε is the bandwidth.

Approximating functions from data

Nonlinear functions

Consider special functions ϕ, so-called *radial basis functions* [Schölkopf and Smola, 2018]:

$$\phi_l(x) = h(r) = \exp\left(-r^2/\varepsilon^2\right), \; r := \|x_l - x\|. \tag{6}$$

The point x_l is the center of the basis function (where it attains its maximum value 1), and the parameter ε is the bandwidth.

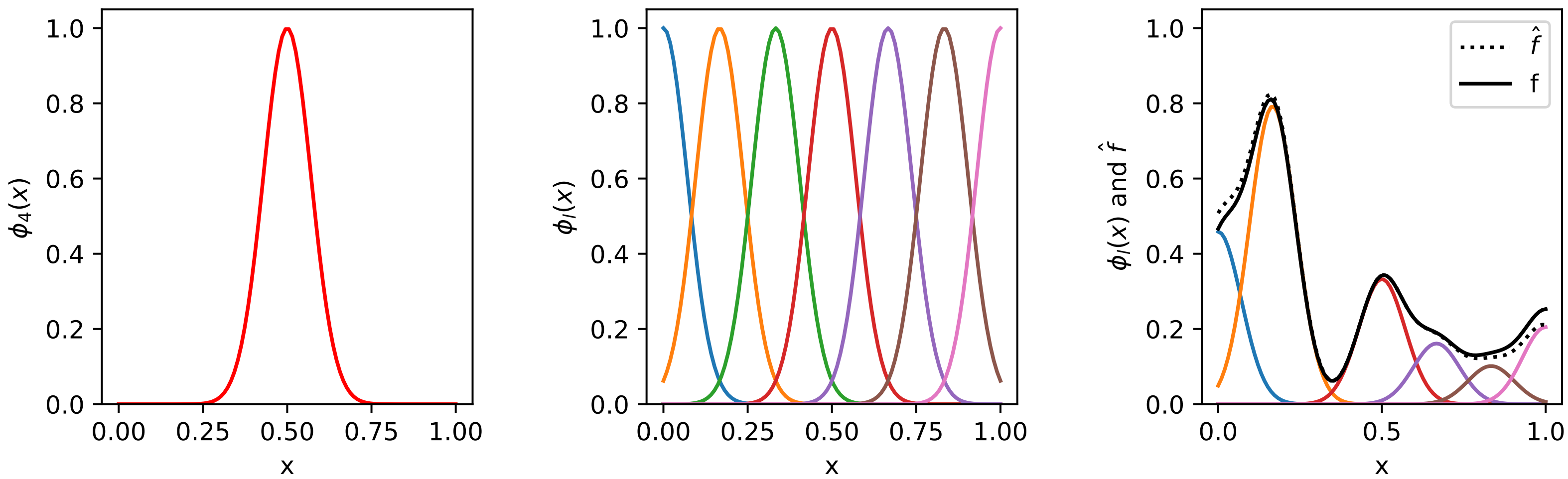


Figure: Left panel: a single radial basis function ϕ_4 with $x_4 = 0.5$ and $\varepsilon = 0.1$. Center panel: radial basis functions ϕ_l. Right panel: approximation of a function f as a sum $\hat{f}$. Coefficient values in the approximation $\sum_{l=1}^{7} c_l \phi_l(x) = \hat{f}(x)$: $c_l = 4.58e-01,\ 7.92e-01,\ 1.25e-04,\ 3.32e-01,\ 1.61e-01,\ 1.01e-01,\ 2.04e-01$.

Side note

Equivalence to neural networks

Kernel methods (in particular: Gaussian processes [Rasmussen and Williams, 2005]) are equivalent to infinitely wide shallow neural networks [Neal, 1996] and infinitely wide, finitely deep neural networks [Lee et al., 2018].

Core idea

Think of normally distributed weights in an **untrained** neural network. Then the rescaled average of them, after the nonlinear activation function, is again a normally distributed variable (by the central limit theorem below). The **trained** network can then be modelled as the posterior distribution of a Gaussian process.

If $X_1, X_2, \dots$ are random samples each of size n taken from a population with overall mean μ and finite variance σ^2 and if $\bar{X}_n$ is the sample mean, the limiting form of the distribution of $Z = \sqrt{n}\frac{\bar{X}_n - \mu}{\sigma}$ as $n \to \infty$, is the standard normal distribution.

Amazingly, we can analytically compute the kernel functions for several neural network nonlinearities!

Vector fields from observations

The most important message of the lecture(s)

Observation is all there is - you can never obtain the underlying state of a system.

Vector fields from observations

The most important message of the lecture(s)

Observation is all there is - you can never obtain the underlying state of a system.

It is impossible to obtain a full representation of the system state for a naturally occuring system. The only data you can gather are sensor observations, mostly in the form of real-valued data: images, temperature, pressure, ...

Vector fields from observations

The most important message of the lecture(s)

Observation is all there is - you can never obtain the underlying state of a system.

Vector fields from observations

The most important message of the lecture(s)

Observation is all there is - you can never obtain the underlying state of a system.

How do we go from here? Natural science!

1. Choose what you want to observe, what you are interested in!
2. Then think of models and build systems that can recreate that.
3. [Optional] Call your models “reality”, because that is what humans do.

Vector fields from observations

The most important message of the lecture(s)

Observation is all there is - you can never obtain the underlying state of a system.

How do we go from here? Natural science!

1. Choose what you want to observe, what you are interested in!
2. Then think of models and build systems that can recreate that.
3. [Optional] Call your models “reality”, because that is what humans do.

How is this useful in machine learning?

1. We can now observe many things that interest us, in the form of real-valued sensor data.
2. Machines are now much better than humans when it comes to evaluating massive amounts heterogeneous sensor data.
3. We can use machines to recreate the important parts of the underlying systems!

Vector fields from observations

Recap: What is a dynamical system?

“The notion of a dynamical system is the mathematical formalization of the general scientific concept of a deterministic process.” [Kuznetsov, 2004]

A dynamical system is a triple (T, X, ψ), with

- the state space X (Euclidean space $\mathbb{R}^n$, manifold $\mathscr{M}$, metric space,...),
- the time T (continuous $\mathbb{R}$, $\mathbb{R}_0^+$, discrete $\mathbb{Z}$, $\mathbb{N}$, ...), and
- the evolution operator $\psi : T \times X \to X$, with the following properties for all $x \in X$:

P1 $\psi(0, x) = \mathrm{Id}(x) = x$,

P2 $\psi(t + s, x) = \psi(t, \psi(s, x)) = (\psi_t \circ \psi_s)(x)$ for all $t, s \in T$.

Vector fields from observations

Vector fields describing the evolution operator

On a manifold $\mathscr{M}$, the tangent bundle is the disjoint union of the tangent spaces at every point,

$$T\mathscr{M} := \cup_{x\in\mathscr{M}} T_x\mathscr{M}. \tag{7}$$

A vector field v is a section of the tangent bundle: $v : \mathscr{M} \to T\mathscr{M}$, such that $x \mapsto v(x) \in T_x\mathscr{M}$. That means the vector field assigns every point on the manifold a vector in the local tangent space.

With this prerequisite, an evolution operator ψ of a dynamical system on a manifold can be defined implicitly and without coordinates through

$$\left.\frac{d\psi}{dt}\right|_{t=0}(x) = v(x). \tag{8}$$

Note that $v(x) \in T_x\mathscr{M}$, the vectors are not elements of $\mathscr{M}$!

However: $T\mathbb{R}^n \simeq \mathbb{R}^n$.

Vector fields from observations

Vector fields are functions

1. A vector field assigns each point x in the space $\mathscr{M}$ a vector in the tangent space at that point.

Vector fields from observations

Vector fields are functions

1. A vector field assigns each point x in the space $\mathscr{M}$ a vector in the tangent space at that point.
2. You have already plotted vector fields in exercise three, to visualize dynamical systems through their phase portraits.

Vector fields from observations

Vector fields are functions

1. A vector field assigns each point x in the space $\mathscr{M}$ a vector in the tangent space at that point.
2. You have already plotted vector fields in exercise three, to visualize dynamical systems through their phase portraits.
3. If the space $\mathscr{M}$ is just some Euclidean space $\mathbb{R}^n$, the tangent spaces $T_x\mathscr{M}$ are usually identified with the same Euclidean space, such that a vector field turns into a function $v : \mathbb{R}^n \to \mathbb{R}^n$.

Vector fields from observations

Vector fields are functions

1. A vector field assigns each point x in the space $\mathscr{M}$ a vector in the tangent space at that point.
2. You have already plotted vector fields in exercise three, to visualize dynamical systems through their phase portraits.
3. If the space $\mathscr{M}$ is just some Euclidean space $\mathbb{R}^n$, the tangent spaces $T_x\mathscr{M}$ are usually identified with the same Euclidean space, such that a vector field turns into a function $v : \mathbb{R}^n \to \mathbb{R}^n$.
4. This is great, because you know how to approximate such functions!

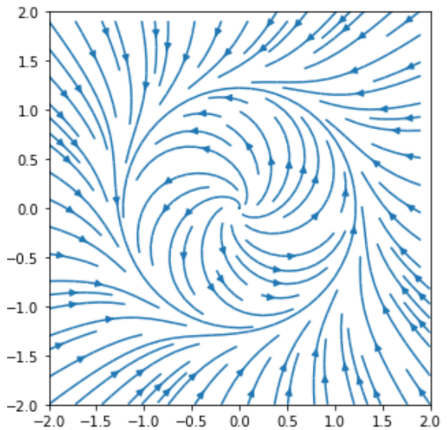


Figure: Vector field and streamlines for the limit cycle in the Andronov-Hopf normal form.

Time-delay embedding

The most important message of the lecture(s)

Remember: observation is all there is - you can never obtain the underlying state of a system.

Time-delay embedding

The most important message of the lecture(s)

Remember: observation is all there is - you can never obtain the underlying state of a system.

Question: how can you approximate $f(x)$ if you do not even know the state x?

Time-delay embedding

The most important message of the lecture(s)

Remember: observation is all there is - you can never obtain the underlying state of a system.

Question: how can you approximate $f(x)$ if you do not even know the state x?

Now [Takens, 1981] writes: at least you can get *a* state that is predictive for your observations!

Time-delay embedding

Illustration

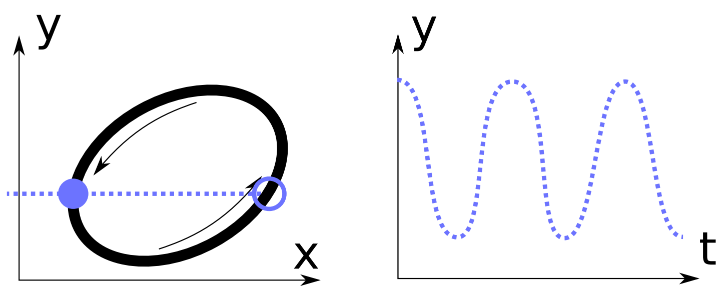


Figure: Concept of scalar-valued observations of a higher-dimensional system. A point (filled circle) is moving on a circle (left) in two dimensions, but only the y-coordinate is measured over time. If only a single observation $y(t)$ is known, one cannot tell exactly where the point is, or in which direction it is moving.

Time-delay embedding

Illustration

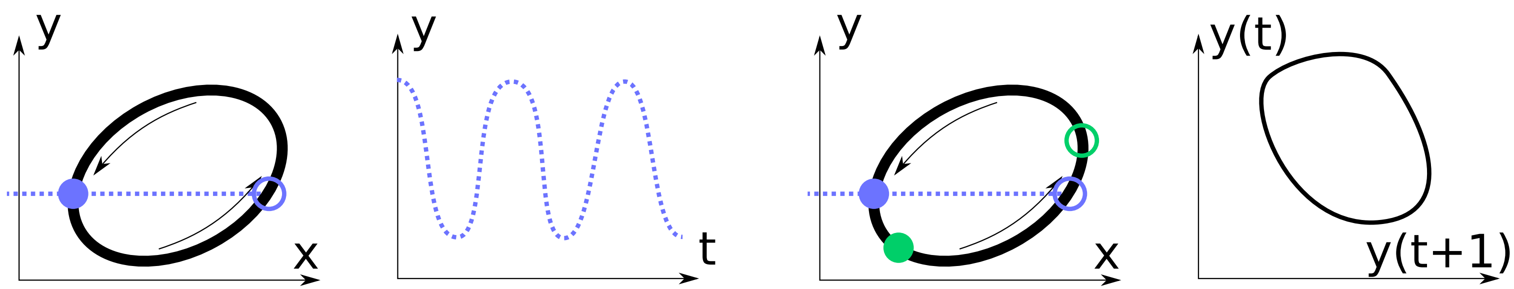


Figure: Concept of scalar-valued observations of a higher-dimensional system. A point (filled circle) is moving on a circle (left) in two dimensions, but only the y-coordinate is measured over time. If only a single observation $y(t)$ is known, one cannot tell exactly where the point is, or in which direction it is moving.

Takens: at least you can get a representation of the state space by using delays of y!

Time-delay embedding

Illustration

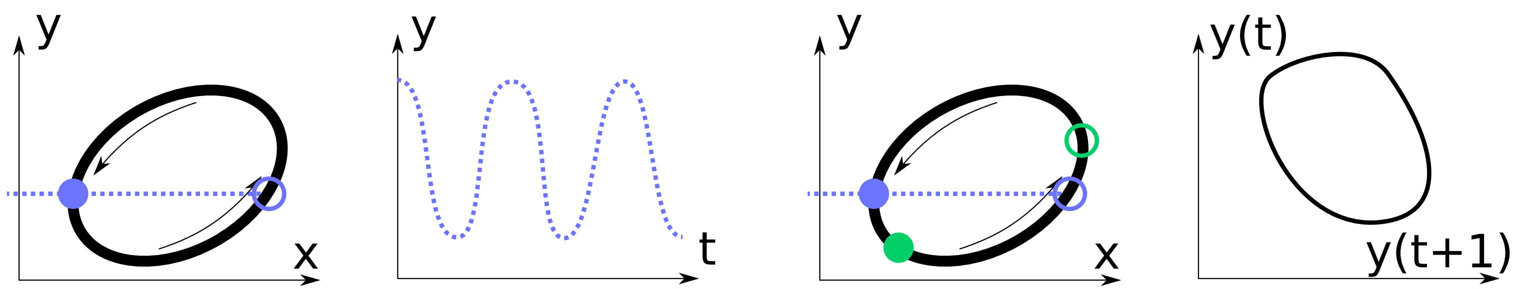


Figure: Concept of scalar-valued observations of a higher-dimensional system. A point (filled circle) is moving on a circle (left) in two dimensions, but only the y-coordinate is measured over time. If only a single observation $y(t)$ is known, one cannot tell exactly where the point is, or in which direction it is moving.

Takens: at least you can get a representation of the state space by using delays of y!
With this representation, you can tell where the point is moving.

Time-delay embedding

Creating state from observation: Takens' delay embedding

Let $k \geq d \in \mathbb{N}$, and $\mathscr{M} \subset \mathbb{R}^k$ be a d-dimensional, compact, smooth, connected, oriented manifold with Riemannian metric g induced by the embedding in k-dimensional Euclidean space.

Time-delay embedding

Creating state from observation: Takens' delay embedding

Let $k \geq d \in \mathbb{N}$, and $\mathscr{M} \subset \mathbb{R}^k$ be a d-dimensional, compact, smooth, connected, oriented manifold with Riemannian metric g induced by the embedding in k-dimensional Euclidean space.
Note: think of ψ as an evolution operator and y as an observation function!

Theorem

Generic delay embeddings. *For pairs (ψ, y), $\psi : \mathscr{M} \to \mathscr{M}$ a smooth diffeomorphism and $y : \mathscr{M} \to \mathbb{R}$ a smooth function, it is a generic property that the map $E_{(\psi,y)} : \mathscr{M} \to \mathbb{R}^{2d+1}$, defined by*

$$E_{(\psi,y)}(x) = \left(y(x), y(\psi(x)), \dots, y(\underbrace{\psi \circ \cdots \circ \psi}_{2d \text{ times}}(x)) \right)$$

is an embedding of $\mathscr{M}$; here, "smooth" means at least C^2.

Time-delay embedding

Creating state from observation: Takens' delay embedding

Let $k \geq d \in \mathbb{N}$, and $\mathscr{M} \subset \mathbb{R}^k$ be a d-dimensional, compact, smooth, connected, oriented manifold with Riemannian metric g induced by the embedding in k-dimensional Euclidean space.

Note: think of ψ as an evolution operator and y as an observation function!

Theorem

Generic delay embeddings. *For pairs (ψ, y), $\psi : \mathscr{M} \to \mathscr{M}$ a smooth diffeomorphism and $y : \mathscr{M} \to \mathbb{R}$ a smooth function, it is a generic property that the map $E_{(\psi,y)} : \mathscr{M} \to \mathbb{R}^{2d+1}$, defined by*

$$E_{(\psi,y)}(x) = \left(y(x), y(\psi(x)), \ldots, y(\underbrace{\psi \circ \cdots \circ \psi}_{2d \text{ times}}(x)) \right)$$

is an embedding of $\mathscr{M}$; here, "smooth" means at least C^2.

Note: genericity here means the set of candidates is "open and dense" in a function space.

Summary

Extracting dynamical systems from data

1. Approximation of functions
2. Vector fields
3. Takens time-delay embedding

Literature I

Kuznetsov, Y. A. (2004).
Elements of Applied Bifurcation Theory.
Springer New York.

Lee, J., Sohl-dickstein, J., Pennington, J., Novak, R., Schoenholz, S., and Bahri, Y. (2018).
Deep Neural Networks as Gaussian Processes.
In *International Conference on Learning Representations*, pages 1–17.

Neal, R. M. (1996).
Priors for Infinite Networks.
In *Bayesian Learning for Neural Networks*, pages 29–53. Springer New York.

Rasmussen, C. E. and Williams, C. K. I. (2005).
Gaussian Processes for Machine Learning (Adaptive Computation And Machine Learning).
The MIT Press.

Schölkopf, B. and Smola, A. J. (2018).
Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.
The MIT Press.

Takens, F. (1981).
Detecting strange attractors in turbulence.
Lecture Notes in Mathematics, pages 366–381.