

Fundamental concepts

Common time-series data types in neuroscience

- Intracellular voltage recordings
- Spike trains
- LFP/ECOG
- Calcium imaging
- fMRI/EEG/MEG
- Fiber photometry
- Video/tracked behavior
- Stimuli/other sensors
- Simulated data

Challenges of time-series analysis

Discriminative vs generative models

Model fitting

- Parameters
- Loss functions
- Training/test data
- Bias-variance tradeoff

Model comparison

Random processes perspective

Dynamical systems perspective

Common types of time-series data in neuroscience (I)

Electrical recordings of neural activity

Intracellular voltage

OP-amp+resistor
connected to
electrode

Voltage trace of
action potential

- > Direct access to membrane potential
- > Very hard *in vivo*
- > Can record few neurons at a time
- > Can easily see APs
- > High temporal resolution

Extracellular voltage/spike trains

Electrode near a
bunch of neurons

2-3 extracellular
voltage traces w
spikes



Spike sort

Raster plots

- > No direct access to membrane potential
- > Common *in vivo* approach
- > Need to “spike sort” to get APs
- > Can record many neurons simultaneously
- > Usually low spatial resolution
- > High temporal resolution
- > Spike trains ~ “point process”

Remove
spikes

LFP

LFP voltage
traces

ECoG

ECoG
apparatus

ECoG
signal

- > Often in humans with epilepsy
- > Average over many neurons
- > Low spatial resolution
- > High temporal resolution

Common types of time-series data in neuroscience (II)

Neuroimaging

Calcium imaging

Mouse w
head-mounted
microscope

Ca-imaging still

Ca-imaging
segmented

3 Ca fluorescence
signals

- > Common *in vivo* approach
- > High spatial resolution
- > Low temporal resolution (~500 ms)
- > Can identify individual neurons
- > Can “sort of” infer spikes
- > Subject to motion artifacts

fMRI

Person in fMRI

Human
brain img w
highlighted
areas

Example
BOLD
timecourse

- > Noninvasive/human-friendly
- > BOLD signal
- > Average over many neurons
- > High spatial resolution
- > Low temporal resolution

EEG/MEG

Person in EEG

Person in MEG

EEG
signals

MEG signals

- > Noninvasive/human-friendly
- > Low spatial resolution (~max 256 channels)
- > Average over MANY neurons
- > High temporal resolution

Common types of time-series data in neuroscience (III)

Behavior

Audio

Zebra finch
spectrogram

Zebra finch

- > High sample rate
- > Often represented as spectrogram
- > Can also cluster/segment

Raw video

Mouse freely
moving video
still

Fly-on-ball
Video still

- > Temporal resolution of camera
- > Useful for neuroethology (characterizing behavior before seeking neural mechanisms)
- > Hard to process directly

Tracked behavior

Flies w tracked
keypoints still

Time-series of
keypoints

- > Usually constructed w computer vision algorithms (DeepLabCut, SLEAP)
- > Often easier to work with than raw video

State/syllable sequences

Ethogram w
segmented behave
and mice
underneath
illustrating behavs

- > Constructed by clustering/segmenting behavioral motifs/syllables
- > Discrete data type

Common types of time-series data in neuroscience (IV)

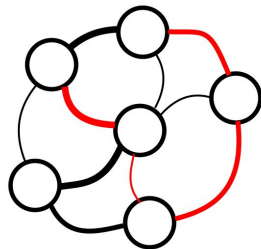
Simulation data

Neural activity: firing rates

Chaos RNN firing
rates over time

> Useful abstraction for
recurrent neural networks

Synaptic strengths



Auxiliary data

$$\tau \frac{dv}{dt} = -v + I(t)$$

if $v > v_{th}$: spike and reset

Neural activity: voltage/spikes

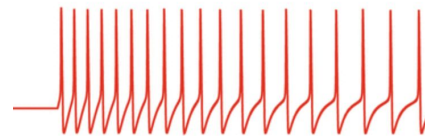
LIF timecourse



> Model of realistic neural responses
> Arbitrary spatial/temporal resolution



> Extremely hard to measure
experimentally



v_{th}



> Useful for predicting behavior of
unobserved variables

Common types of time-series data in neuroscience (V)

Miscellaneous

Stimuli/other sensor data

Primate-task-
style diagonal
stim sequence

Binary TTL-like
signals

- > Used to retrieve “actual times” rather than computer-programmed signals
- > Typically high precision
- > Requires calibration/sync with other devices

Voltage imaging

Voltage
imaging video
still

Voltage +
fluorescence
signal
timecourses

- > Relatively new
- > Often toxic
- > In principle allows direct access to membrane voltage for indiv cells
- > In principal high spatial and temporal resolution

Inferred variables

Population
raster plot w
clear
structure

Scalar
variable
indicating
latent var

- > Constructed from empirical data or simulations
- > Often captures “low-dimensional” signal

Data type by mathematical structure

Univariate

Point-process

Spike trains
Event times

Symbolic/token

Behavioral state
sequence

Continuous

LFP/ECoG recording
Fluorescence/imaging
signal
Firing rate
Voltage
Microphone signal

Multivariate “structured”

Neuroimaging movie (e.g. fluorescence,
BOLD signal, EEG, ECoG)
Behavior video
Tracked behavior keypoints
Stimulus video
Audio spectrogram
Firing rates of spatially arranged neurons

Multivariate

Point-process

Population spike trains

Continuous

Population fluorescence
Multi-channel LFP
Population firing rates
Stimulus patterns

Multi-modal

Simultaneous neural recordings +
behavioral audio/video/tracked keypoints
Stimulus patterns + neural recordings +
behavior
Etc.

Mathematical models

Simplified descriptions of
system/process underlying data

Various types (descriptive, mechanistic...)

“All models are wrong. Some are useful.”



Why are models useful?

Demonstrate self-consistency
of understanding

Test hypotheses against data

Extract interpretable variables/
parameters from data

Can perform experiments on
them *in silico*.

Make
predictions

Discriminative vs generative models

Discriminative models

Real data example 1



Class A

Real data example 2



Class B

Input = data

Output = labels, clusters, segments, etc.

No way to “sample” artificial inputs

Generative models

Artificial sample 1



Artificial sample 2



$P(D|\theta)$



Can *generate* artificial data
in same format as input data.

Can be used as discriminative models.

Model fitting

Parameters

Loss functions

Training/validation/test data

Bias-variance tradeoff

Overfitting

Model comparison

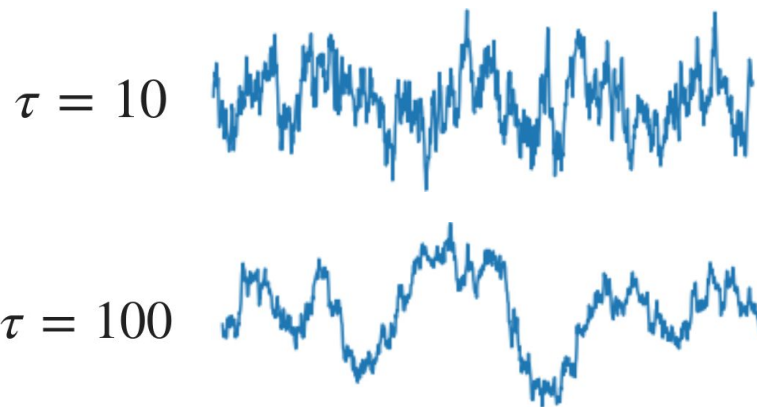
Model fitting - Parameters

Models specified by equation and *parameters*

$$\text{Equation } \left\{ \frac{dx}{dt} = \frac{-x}{\tau} + u(t) \right. \quad \swarrow \text{Parameter}$$

$$\theta \equiv \{\text{param}_1, \text{param}_2, \dots\}$$

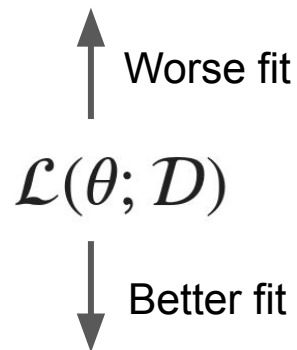
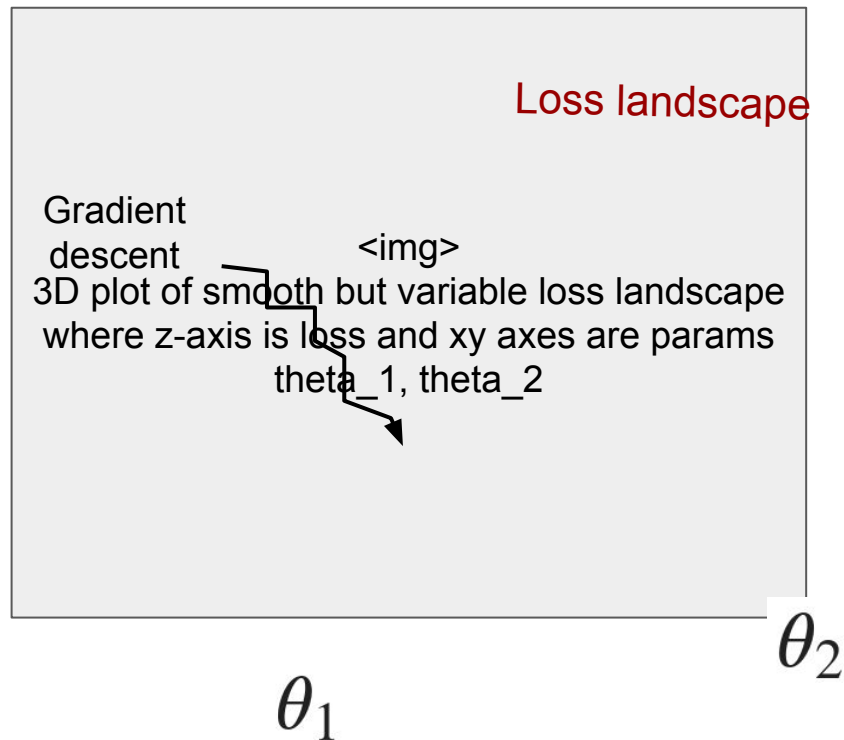
Different parameters yield different behavior



Complexity/flexibility \sim number of parameters

$$\frac{d\mathbf{x}}{dt} = \frac{-\mathbf{x}}{\tau} + \mathbf{J} \tanh(\mathbf{x}) + \mathbf{u}(t) \quad \mathbf{J} = \begin{bmatrix} J_{11}, \dots, J_{1N} \\ \vdots \\ J_{N1}, \dots, J_{NN} \end{bmatrix} \quad N^2+1 \text{ parameters}$$

Model fitting - Loss functions & gradient descent



Deterministic Loss

“Distance” from model predictions to data.

Probabilistic Loss

E.g. $-P(\text{data}|\theta)$

Regularizer

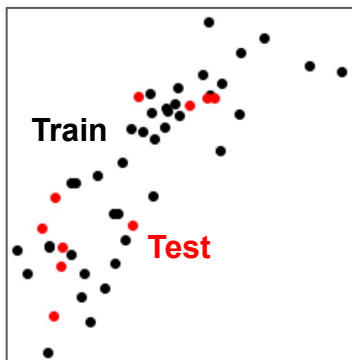
$$\mathcal{L}(\theta; \mathcal{D}) = \mathcal{L}_0(\theta; \mathcal{D}) + \mathcal{L}^*(\theta)$$

Dataset

Model fitting - Training/test data

Often want model to predict never-before-seen data

1. Fit model to **training** data.
2. Evaluate generalization performance on held-out **test** data



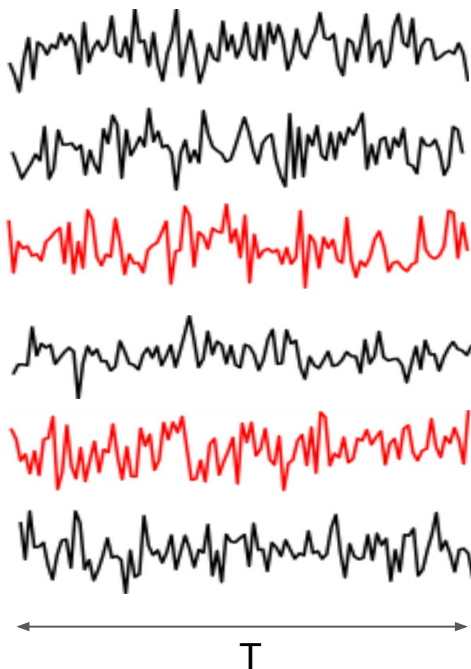
Training & **test** data should be statistically independent given model

Often have **training/validation/test** data.

- **Validation** data used to simulate generalization to **test** data.
- Should only use **test** data for final performance evaluation.

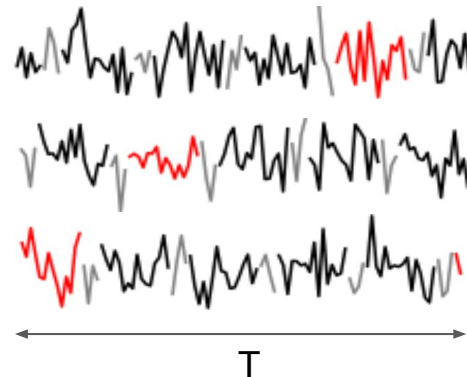
Time-series Case 1:

Train/test → different “trials”



Time-series Case 2:

Train/test → different timepoints



Train & **test** points should be separated well beyond autocorrelation time* of signal (otherwise not independent)

**Need to estimate from data (can be challenging when signals have long timescales)*

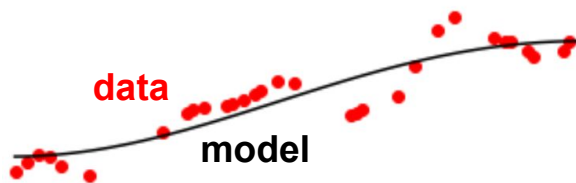
Usually average analysis results over N random **train/test** splits

Model fitting - Bias-variance tradeoff

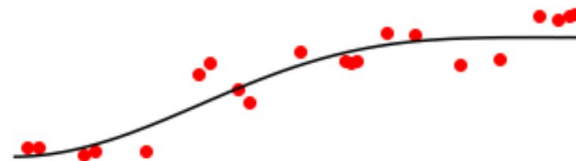
Low-variance, high-bias
(usually fewer parameters)

High-variance, low-bias
(usually more parameters)

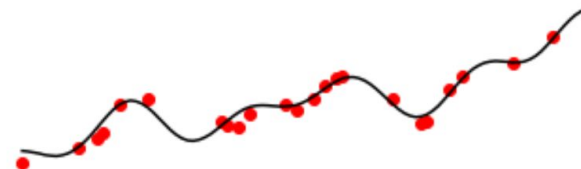
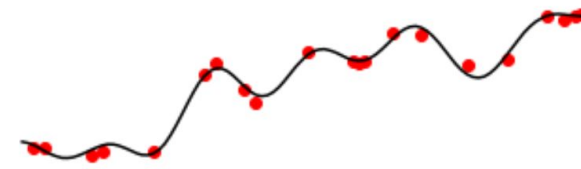
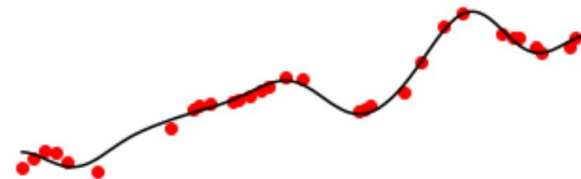
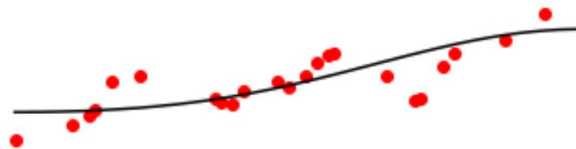
Experiment 1



Experiment 2



Experiment 3



(*but see “double descent”)

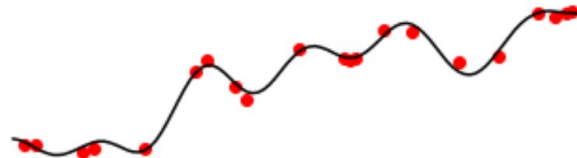
Overfitting

Typically think of data as signal + noise: $y_i = f(x_i) + \eta$

During fitting, want model to distinguish signal from noise

But when # parameters \sim # data points \Rightarrow model can treat noise as signal

- Fits training data near-perfectly
- Usually generalizes poorly to held-out data



Common check: check goodness-of-fit on *held-out data* not used in training (“validation” set)

- Validation set should be *independent* samples from training set (non-trivial for time-series)

Solutions:

- Use fewer parameters (i.e. “simpler” model)
- Use more data (decrease #params / #data)
- Regularize parameters (introduce penalty to loss function e.g. that keeps params small)
- Add *more* parameters (“double descent” phenomenon in modern ML)

Model fitting - Parameters vs hyperparameters

Typically

Parameters → “Knobs” of model to turn

- Fit during “inner loop” e.g. gradient descent

Hyperparameters → model “architectural features”

- Determined in “outer loop” e.g. grid search

Usually # parameters \gg # hyperparameters

Example 1: Linear filter

Filter parameterized by $h(0), h(1), \dots, h(T)$.

Hyperparameter = T (filter length).

Example optimization routine:

For each $T = T_1, \dots, T_N$: fit $h(0:T)$ to data

Select T with best goodness-of-fit

Example 2: Artificial neural network

Parameters = weights

Hyperparameters = # layers, learning rate, ...

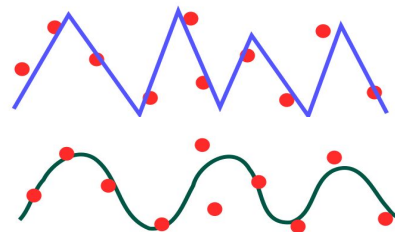
Model comparison

Can compare across parameters, hyperparameters, model classes

- “Model comparison” ~ usually comparing best-fit models given different hyperparameters or classes

“Best” model is *subjective*

- Typical “score” → goodness-of-fit (+ penalty)
- Different models can explain different features of data



Various quantitative ways to score models:

- Information criteria (e.g. AIC/BIC): goodness-of-fit penalized by # params
 - Models with more parameters usually more flexible (higher variance)
 - All data used to fit model
- Cross-validation: goodness-of-fit on held-out data
 - Training data → Used to fit model
 - Validation/test data → Used to eval model
 - Requires more data (since fit only uses X% data)

Nested model analysis

Model A: $\theta = (\theta_1, \dots, \theta_P)$

Model A': $\theta = (\theta_1, \dots, \theta_P, \theta_{P+1})$

- E.g. θ_{P+1} = weight on feature P+1

If $\text{Loss}(A') < \text{Loss}(A)$: include θ_{P+1} in model

Common use case: Can we predict $z(t)$ better from $x(t)$ & $y(t)$ than from just $x(t)$?

Multiple models can explain data equally well

Dataset generated via *Random Process*
(a.k.a *Stochastic Process*)

$$x(t) \sim P[x(t)]$$

Distribution over
trajectories $x(t)$

$$\mathcal{D} \equiv \{x_1(t), \dots, x_n(t)\}$$

Dataset of n sample trajectories

Random processes perspective (I)

Univariate probability
distribution

$P(x)$

*short for $P(X=x)$

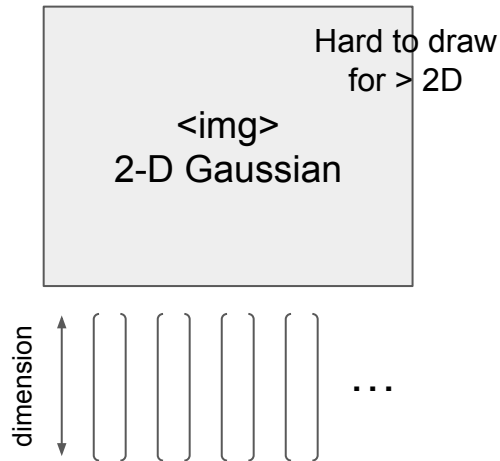


3.13, 2.5, -2.62, 10.23, ...

Samples \rightarrow numbers

Multivariate probability
distribution

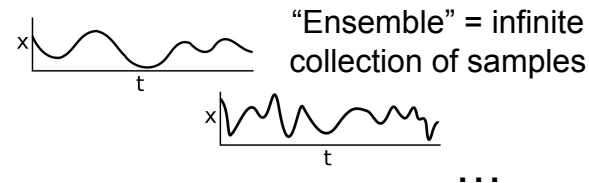
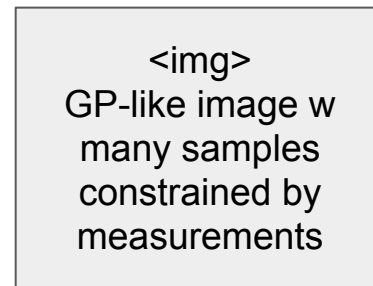
$P(\mathbf{x})$



Samples \rightarrow vectors
(indexed by i : v_i)

Random process

$P[x(t)]$



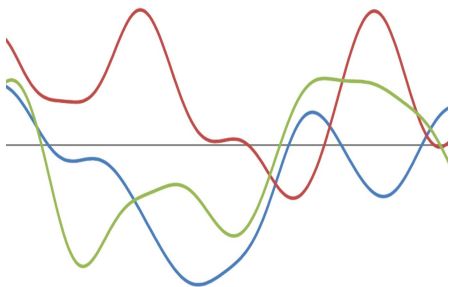
Samples \rightarrow functions
("indexed" by t : $x(t)$)

Equivalent in code
when represented as arrays

Random processes perspective (II)

Common random processes

Gaussian Process



All subsets $[X_1(t), \dots, X_T(t)]$
jointly Gaussian distributed

Specified by mean and
covariance function

Common model of continuous
data with structure

Poisson Process



Spike trains corresponding to
rates above

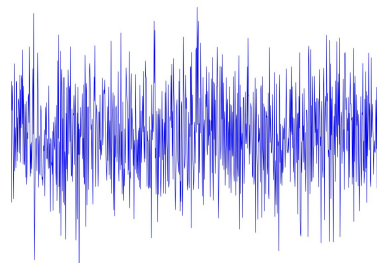
Specified by “rate” $r(t)$

“Point process” (outputs series of
delta functions)

All timepoints $s(t)$ independent
given rate $r(t)$

Common model for spike trains

White noise



All timepoints independent

“Infinite variance”

Can be type of
Gaussian process

Common model of noise

Brownian motion



Integrated white noise

All increments independent

Type of Gaussian
process

Common model of
accumulation process

Random processes perspective (III)

Key probability concepts

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

Joint
distribution

Conditional
distribution

Marginal
distribution

$$P(x(t_1), \dots, x(t_n)) = P(x(t_n)|x(t_1), \dots, x(t_{n-1}))P(x(t_1), \dots, x(t_{n-1}))$$

Chain rule

Bayes' Rule

Also Bayes' Rule

$$\begin{aligned} P(x(t_1), \dots, x(t_n)) = & \\ & P(x(t_n)|x(t_1), \dots, x(t_{n-1})) \\ & \times P(x(t_{n-1})|x(t_1), \dots, x(t_{n-2})) \\ & \times P(x(t_{n-2})|x(t_1), \dots, x(t_{n-3})) \\ & \vdots \end{aligned}$$

$$\begin{aligned} P(x|y) &= \frac{P(y|x)P(x)}{P(y)} \\ P(\theta|D) &= \frac{P(D|\theta)P(\theta)}{P(D)} \end{aligned}$$

$$\begin{aligned} P(\theta|x_t, x_{t-1}, \dots) = & \\ & \frac{P(x_t|\theta, x_{t-1}, \dots)P(\theta|x_{t-1}, \dots)}{P(x_t, x_{t-1}, \dots)} \end{aligned}$$

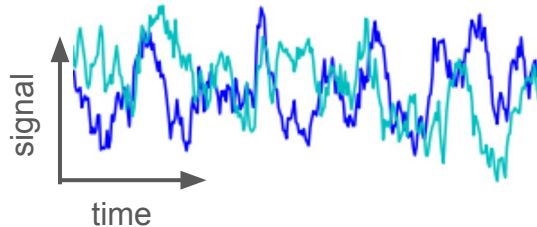
Random processes perspective (IV)

Key random processes concepts: stationarity

Stationarity: Joint statistics don't depend on absolute time, e.g.

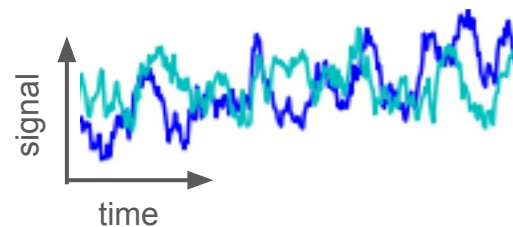
$$P(x(t_1), x(t_1 + \tau)) = P(x(t_2), x(t_2 + \tau))$$

Stationary



*Many models/analyses
assume stationarity of signal*

Not stationary
(increases over time)



(*Could be stationary over longer timescale)

Nonstationary: any statistic can
change over time, not just mean

Random processes perspective (V)

Autocovariance

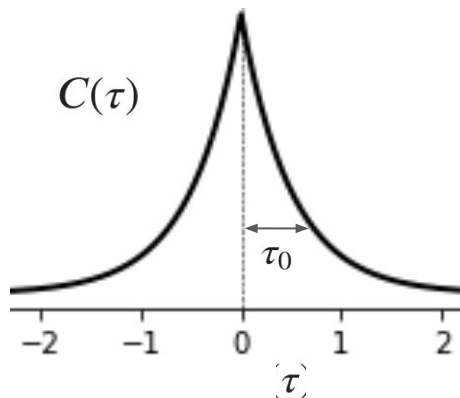
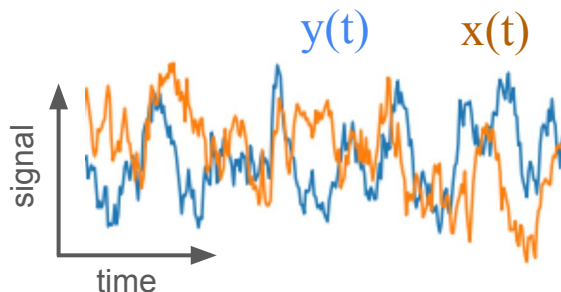
$$C(\tau) = \mathbb{E}[(x(t) - \mathbb{E}[x(t)])(x(t + \tau) - \mathbb{E}[x(t)])]$$

Assumes stationarity
(depends on time lag only)

“Graphical” statistic summarizing how quickly
signal changes over time/how signal at nearby
timepoints are related

Covariance of signal at two
times separated by τ

Property of model $P[x(t)]$
but can estimate from data



“Correlation time” τ_0 describes
timescale of signal fluctuations
(*depends on well-behaved
autocovariance function)

Autocorrelation

$$R(\tau) = \mathbb{E}[x(t)x(t + \tau)]$$

(equivalent to autocovariance
for zero-mean processes)

= Fourier transform of
Power spectral density
(Wiener-Khinchin theorem)

Random processes perspective (VI)

Cross-covariance

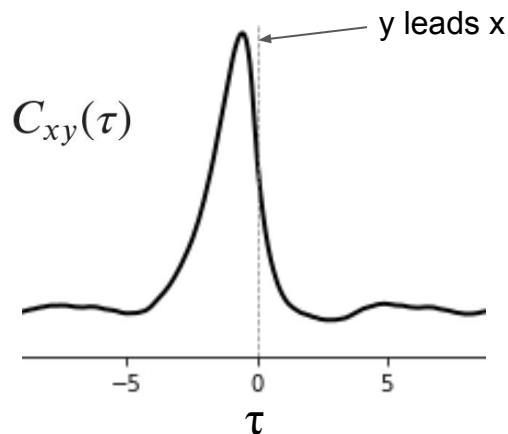
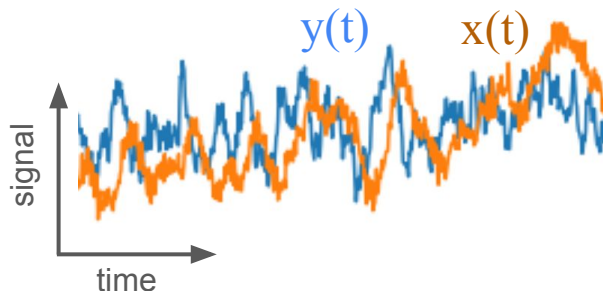
$$C_{xy}(\tau) = \mathbb{E}[(x(t) - \mathbb{E}[x(t)])(y(t + \tau) - \mathbb{E}[y(t)])]$$

Assumes stationarity
(depends on time lag only)

Key statistic describing how two time-series are related

- Useful for detecting leading/lagging processes

Property of model $P[x(t), y(t)]$
but can estimate from data



Generally not symmetric
(unlike autocovariance)

$$R_{xy}(\tau) = \mathbb{E}[x(t)y(t + \tau)]$$

Cross-correlation
(equivalent to
cross-covariance for
zero-mean processes)

= Fourier transform of
Cross-spectral density
(Wiener-Khinchin theorem)

Random processes perspective (VII)

Comparing/fitting random process models to data

Bayes' Rule

$$\text{Posterior} \quad \text{Likelihood} \quad \text{Prior} \\ P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)P(\theta)}{P(\mathcal{D})} \\ \text{Evidence (marginal likelihood)}$$

Fully Bayesian goal:

Compute posterior from data.

(But usually hard because of integral in denominator).

Maximum likelihood (ML)

Maximize

$$P(\mathcal{D}|\theta)$$

with respect to θ .

Maximum a posteriori (MAP)

Maximize

$$P(\theta|\mathcal{D})$$

with respect to θ .

Marginal likelihood

$$P(\mathcal{D}|M_\alpha) = \int d\theta P(\mathcal{D}|\theta, M_\alpha)P(\theta|M_\alpha)$$

- > Requires evaluating data probability under generative model
- > Equivalent to MAP with uniform prior (*except over unbounded spaces)

- > Have to choose prior.
- > Can ignore denominator during optimization

- > Useful for comparing model classes

Random processes perspective (VIII)

Learning and inference

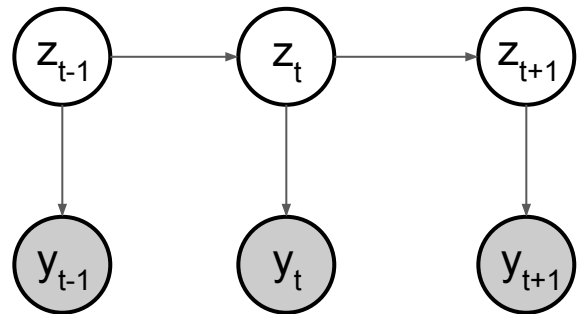
Many models have both “hidden states” + parameters

- Hidden Markov Model (HMM)
- Kalman Filter (like continuous HMM)

Infer hidden states and *learn* parameters

Standard algorithm = Expectation-maximization

- Alternates between inferring hidden states and learning parameters



$$P(y_t | z_t, y_{t-1}, z_{t-1}) = P(y_t | z_t)$$

Generative model of data = *equations of motion*

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x})$$

\mathbf{x} : state

\mathbf{f} : “velocity”

Sample artificial trajectories by integrating velocity over time

Dynamical systems perspective (I)

Dynamical system: set of 1st-order ordinary differential equations*

State space: \mathbb{R}^N

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}) \leftrightarrow \frac{dx_i}{dt} = f_i(x_1, \dots, x_N)$$

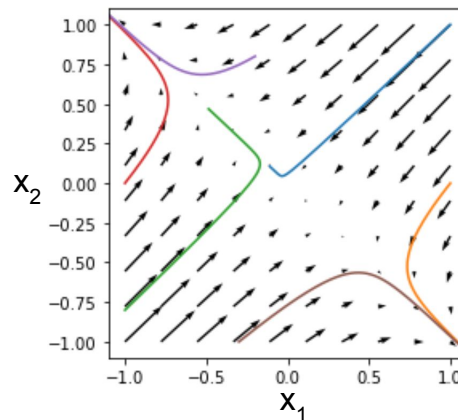
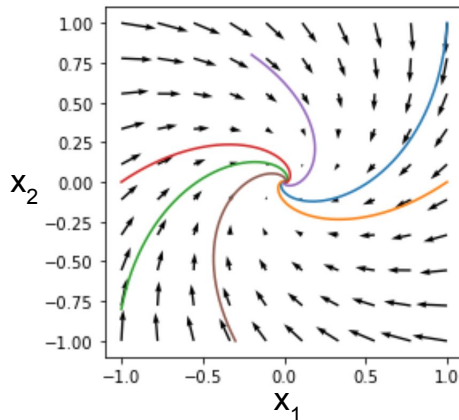
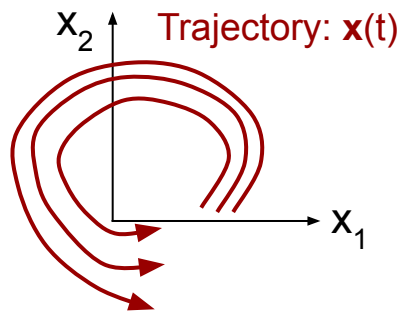
vector notation

component notation

Flow field (also in \mathbb{R}^N)

$$\frac{d\mathbf{x}}{dt} \rightarrow \frac{\Delta\mathbf{x}}{\Delta t}$$

Discrete-time version



Trajectory determined by initial condition $\mathbf{x}(0)$ and flow field
(hard to visualize in $> 3D$)

- Rarely “solve” for $\mathbf{x}(t)$
- Typically: characterize fixed points, timescales, limit cycles, sensitivity to initial conditions, etc

Example inferred
dynamics
trajectories from
some paper

*higher-order systems can be rewritten as 1st-order systems

Dynamical systems perspective (II)

Linear dynamical systems

“Neural” form

$$\frac{d\mathbf{x}}{dt} = \tilde{\mathbf{J}}\mathbf{x} \quad \text{or} \quad \tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + \mathbf{J}\mathbf{x}$$

Specified by single matrix
(+ optional neural timescale τ)

$$\tilde{\mathbf{J}} = \frac{\mathbf{J} - \mathbf{I}}{\tau}$$

Discrete-time version

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} \quad \mathbf{A} = \Delta t \tilde{\mathbf{J}} + \mathbf{I}$$

“Driven” version

$$\tau \frac{d\mathbf{x}}{dt} = -\mathbf{x} + \mathbf{J}\mathbf{x} + \mathbf{B}\mathbf{u}(t)$$

Typically decay to zero or blow up to infinity

- Depends on max eigenvalue of \mathbf{J}
- $\lambda_{\max} < 0 \Rightarrow$ decay, $\lambda_{\max} > 0 \Rightarrow$ explode

Periodic behavior requires “fine-tuning”: $\lambda_{\max} = 0$

At most 1 stable fixed point at origin

Stable limit cycle impossible

Chaos impossible

Often used as local approximations of nonlinear dynamics

Real eigenvalues \rightarrow decay/growth dynamics

Imaginary eigenvalues \rightarrow rotational dynamics

Complex eigenvalues \rightarrow decay/growth/rotational dynamics

Dynamical systems perspective (III)

Recurrent neural networks

Rate RNN

$$\tau \frac{d\mathbf{h}}{dt} = -\mathbf{h} + \mathbf{J}\phi(\mathbf{h}) + \mathbf{B}\mathbf{u}(t)$$

“Voltage” \mathbf{h} Weights \mathbf{J} External inputs $\mathbf{u}(t)$

$$\mathbf{r} \equiv \phi(\mathbf{h})$$

Firing rates

$$\phi(\mathbf{h}) = \tanh(\mathbf{h})$$

Activation function

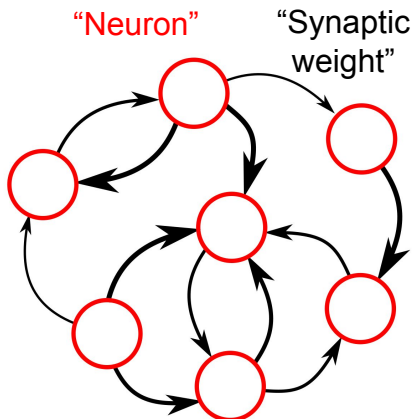
$$= (1 + \tanh(\mathbf{h}))/2$$

$$= \text{ReLU}(\mathbf{h}) \equiv [\mathbf{h}]_+$$

“Nice” (continuous, differentiable)

“Neurons” can model real neurons

OR used as generic flexible model



Spiking RNN

$$\tau \frac{d\mathbf{h}}{dt} = -\mathbf{h} + \mathbf{J}\mathbf{s}(t) + \mathbf{B}\mathbf{u}(t)$$

Spike trains $\mathbf{s}(t)$

$$\text{If } h_i(t) > v_{th}:$$

Emit spike.

Reset to 0 (or v_{reset}) for τ_{RF} .

Not differentiable

Usually model of real neurons

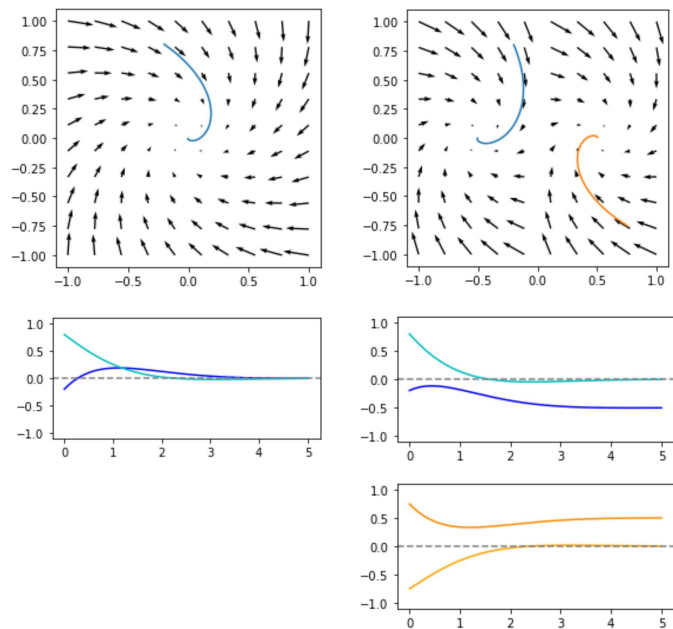
LIF (leaky integrate-and-fire) is most common model

But many variations
(e.g. adaptive LIF)

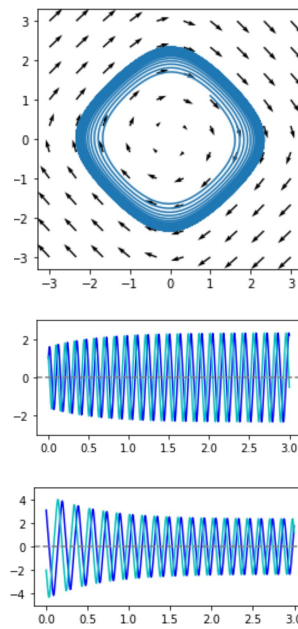
Dynamical systems perspective (IV)

Commonly studied phenomena

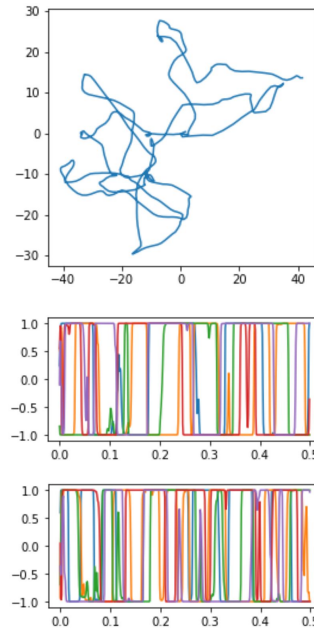
Fixed points



Limit cycles



Chaos



2D projection
(no chaos in 2D)

Common model:
Rate RNN with
large, random
weights and tanh
activation
(Sompolinsky
1988)

Dynamical systems perspective (V)

Noisy dynamical system as a random process

Stochastic
differential
equation (SDE)

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \eta) \longrightarrow P[\mathbf{x}(t)]$$

Noise

$$\frac{d\mathbf{y}}{dt} = \mathbf{g}(\mathbf{x}, \xi) \longrightarrow P[\mathbf{y}(t)|\mathbf{x}(t)]$$

Observation
distribution

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}) + \eta$$

Additive noise
(common)

Variable initial
conditions $\mathbf{x}(0)$ and
inputs $\mathbf{u}(t)$ can also be
source of randomness

Example 1:
 $\mathbf{x}(t)$ = firing rates

Example 2:
 $\mathbf{x}(t)$ = hidden computational variables,
 $\mathbf{y}(t)$ = firing rates

$$P[\mathbf{y}(t)] = \int_{\mathbf{x}(t)} P[\mathbf{y}(t)|\mathbf{x}(t)] P[\mathbf{x}(t)]$$

Dynamical systems perspective (VI)

Comparing/fitting dynamical systems models to data

Approach 1: Qualitatively compare to data

Model system via dynamics equations
Vary parameters to study behavior

Compare fixed points, timescales, etc.

Train RNN to perform task
Then examine RNN dynamics

Use e.g. dimensionality reduction to visualize
empirical vs model dynamics

Approach 2: Fit directly to data

Loss = distance to empirical trajectories
(usually average trajectories over trials)

Loss = negative likelihood (probability* of
trajectories) or a posteriori probability
**Need to add probability/noise
to dynamics model*

**Researchers often model “low-dimensional”
dynamics underlying high-dimensional noisy
neural recordings
(various ways to do this)*