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

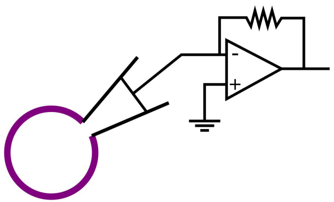
Random processes perspective

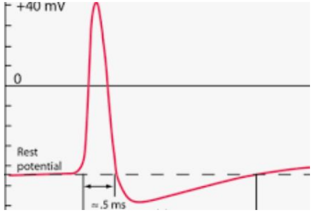
Dynamical systems perspective

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

Electrical recordings of neural activity

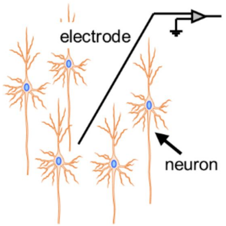
Intracellular voltage

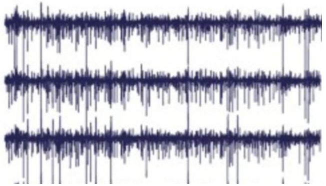




> Direct access to membrane potential
> Very hard *in vivo*
> Can record few neurons at a time
> Can easily see APs
> High temporal resolution

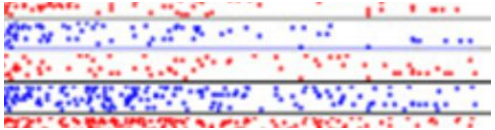
Extracellular voltage/spike trains





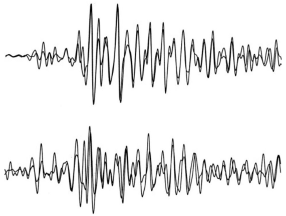
Remove spikes

Spike sort

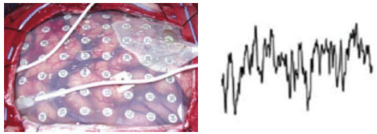


> No direct access to membrane potential
> Common *in vivo* approach
> Need to “spike sort” to get APs
> Can record many neurons simultaneously
> Usually low spatial resolution
> High temporal resolution
> Spike trains ~ “point process”

LFP



ECoG

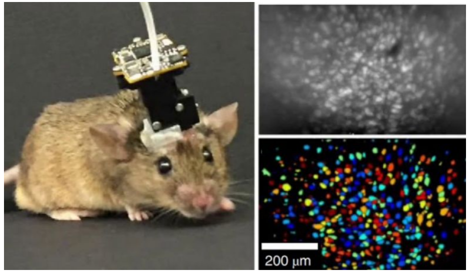


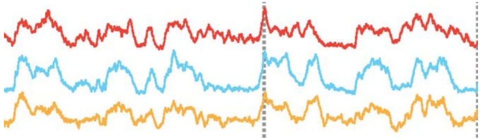
> Often in humans with epilepsy
> Average over many neurons
> Low spatial resolution
> High temporal resolution

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

Neuroimaging

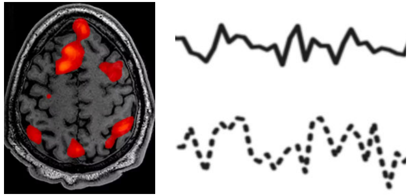
Calcium imaging





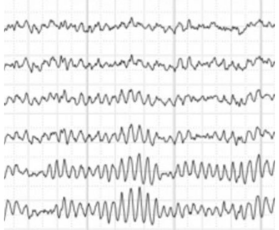
> Common *in vivo* approach
> High spatial resolution
> Low temporal resolution (~500 ms)
> Can identify individual neurons
> Can “sort of” infer spikes
> Subject to motion artifacts

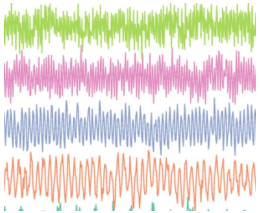
fMRI



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

EEG/MEG



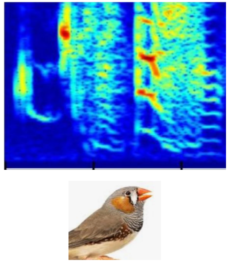


> Noninvasive/human-friendly
> Low spatial resolution
(~max 256 channels)
> Average over MANY neurons
> High temporal resolution

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

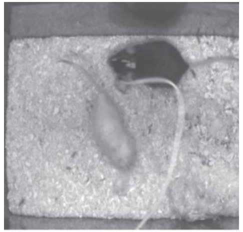
Behavior

Audio



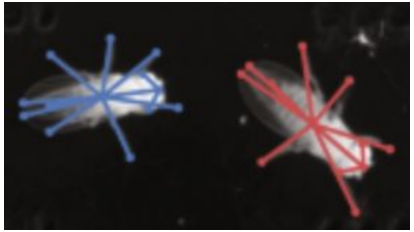
> High sample rate
> Often represented as spectrogram
> Can also cluster/ segment

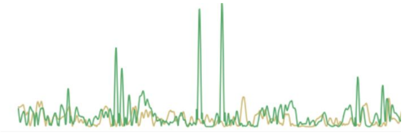
Raw video



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

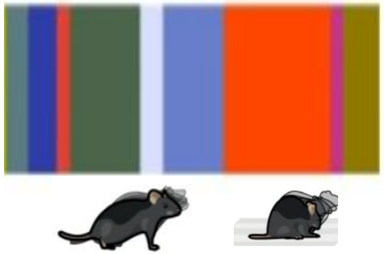
Tracked behavior





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

State/syllable sequences

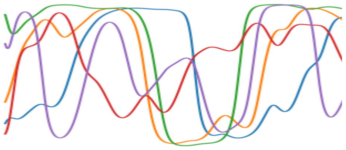


> Constructed by clustering/segmenting behavioral motifs/ syllables
> Discrete data type

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

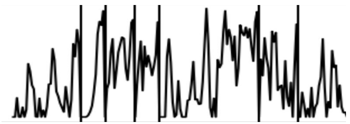
Simulation data

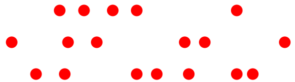
Neural activity: firing rates



> Useful abstraction for recurrent neural networks

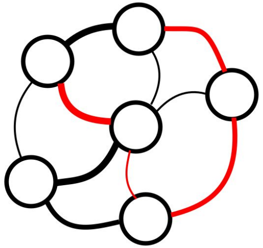
Neural activity: voltage/spikes

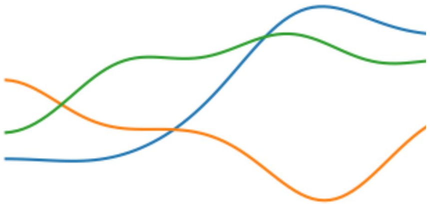




> Model of realistic neural responses
> Arbitrary spatial/temporal resolution

Synaptic strengths



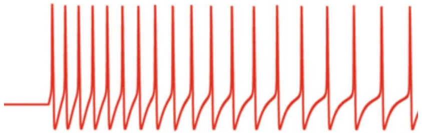


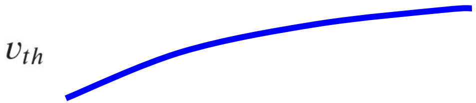
> Extremely hard to measure experimentally

Auxiliary data

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

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



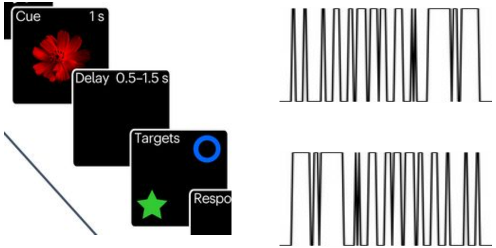


> Useful for predicting behavior of unobserved variables

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

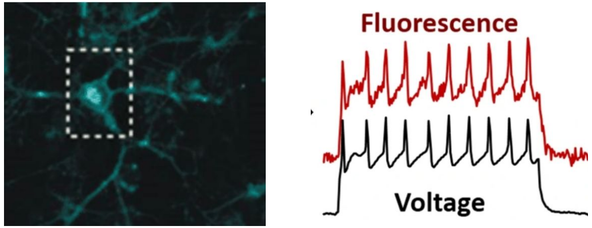
Miscellaneous

Stimuli/other sensor data



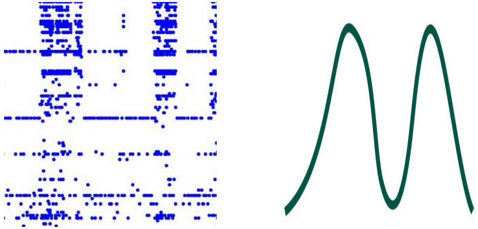
> Used to retrieve “actual times” rather than computer-programmed signals
> Typically high precision
> Requires calibration/sync with other devices

Voltage imaging



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

Inferred variables



> 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

Point-process

Population spike trains

Continuous

Population fluorescence

Multi-channel LFP

Population firing rates

Stimulus patterns

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

Multi-modal

Simultaneous neural recordings + behavioral audio/video/tracked keypoints

Stimulus patterns + neural recordings + behavior

Etc.

Challenges of analyzing time-series data in neuroscience

- Many small decisions to make along the way
- Easy to make mistakes
- Takes time to be rigorous
- *Many* methods to choose from
- Big datasets
- Multi-modal datasets
- Weird statistics/ lack of trials
- Missing data/variable trial lengths
- High dimensionality
- Violates many assumptions of classic signal processing
- Model-fitting can be highly complex
- Difficult to interpret analysis results

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

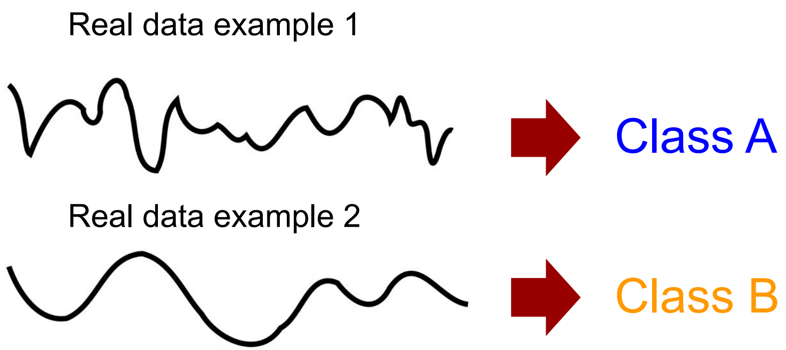
Extract interpretable variables/
parameters from data

Can perform experiments on
them *in silico*.

Make
predictions

Discriminative vs generative models

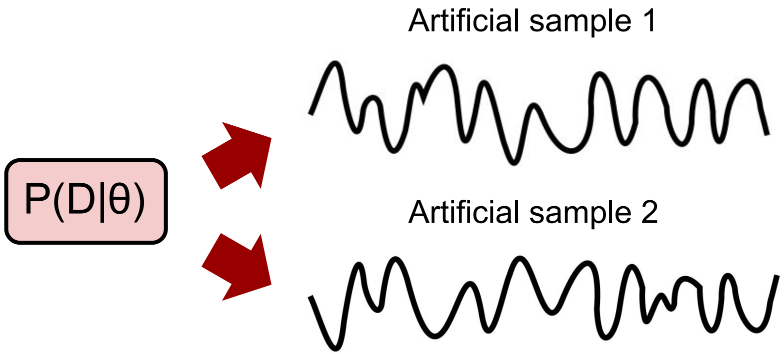
Discriminative models



Input = data
Output = labels, clusters, segments, etc.

No way to “sample” artificial inputs

Generative models



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*

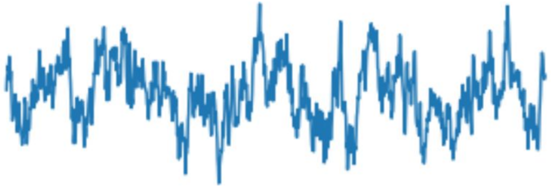
Equation $\left\{ \frac{dx}{dt} = \frac{-x}{\tau} + u(t) \right.$

Parameter

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

Different parameters yield different behavior

$\tau = 10$



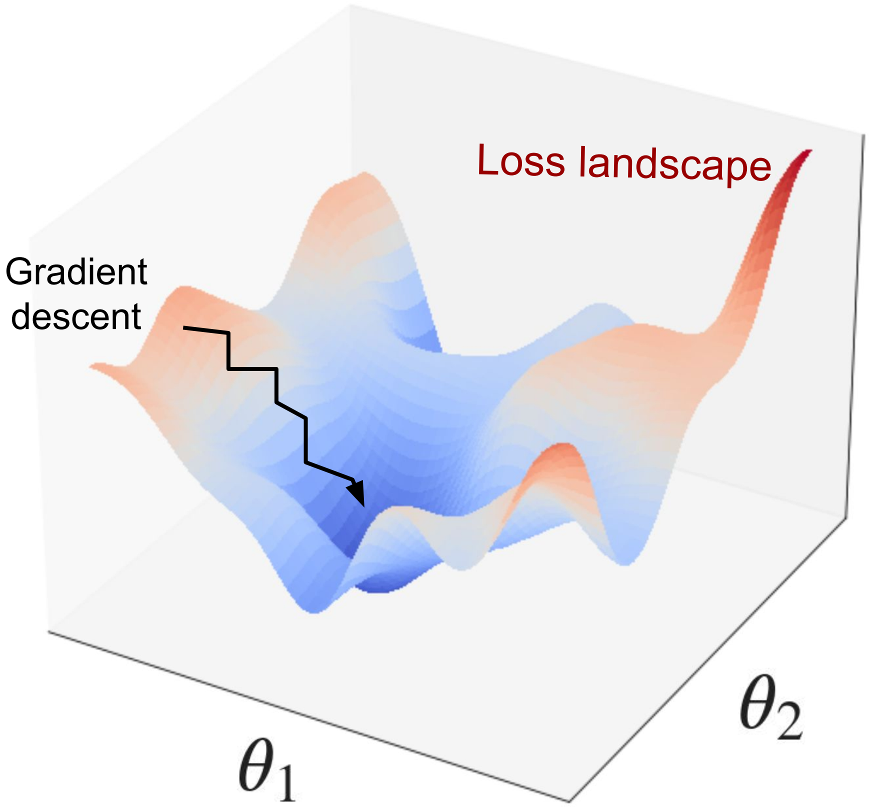
$\tau = 100$

Complexity/flexibility ~ number of parameters

$$\frac{d\mathbf{x}}{dt} = \frac{-x}{\tau} + J\tanh(\mathbf{x}) + \mathbf{u}(t) \qquad J = \begin{bmatrix} J_{11}, \ldots, J_{1N} \\ \vdots \\ J_{N1}, \ldots, J_{NN} \end{bmatrix}$$

N^2+1 parameters

Model fitting - Loss functions & gradient descent



Worse fit

$$\mathcal{L}(\theta; D)$$

Better fit

<u>Deterministic Loss</u>

"Distance" from model predictions to data.

<u>Probabilistic Loss</u>

E.g. -P(data|θ)

Regularizer

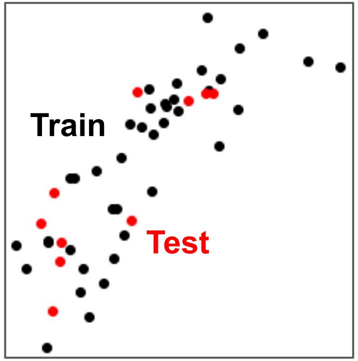
$$\mathcal{L}(\theta; D) = \mathcal{L}_0(\theta; D) + \mathcal{L}^*(\theta)$$

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



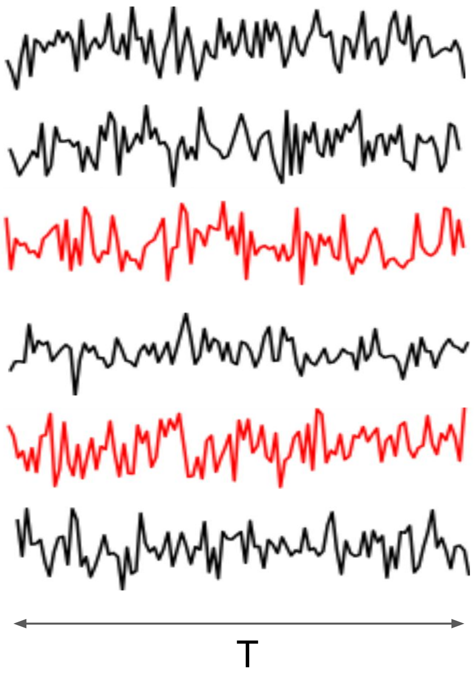
Training & **test** data should be statistically independent given model

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

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

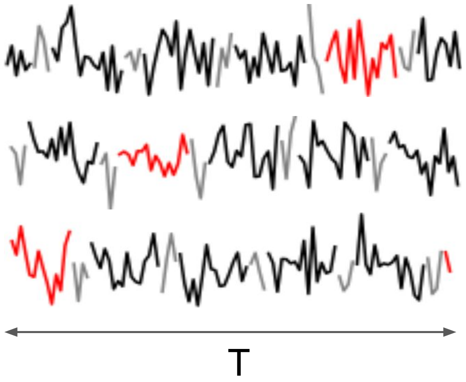
Time-series Case 1:

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

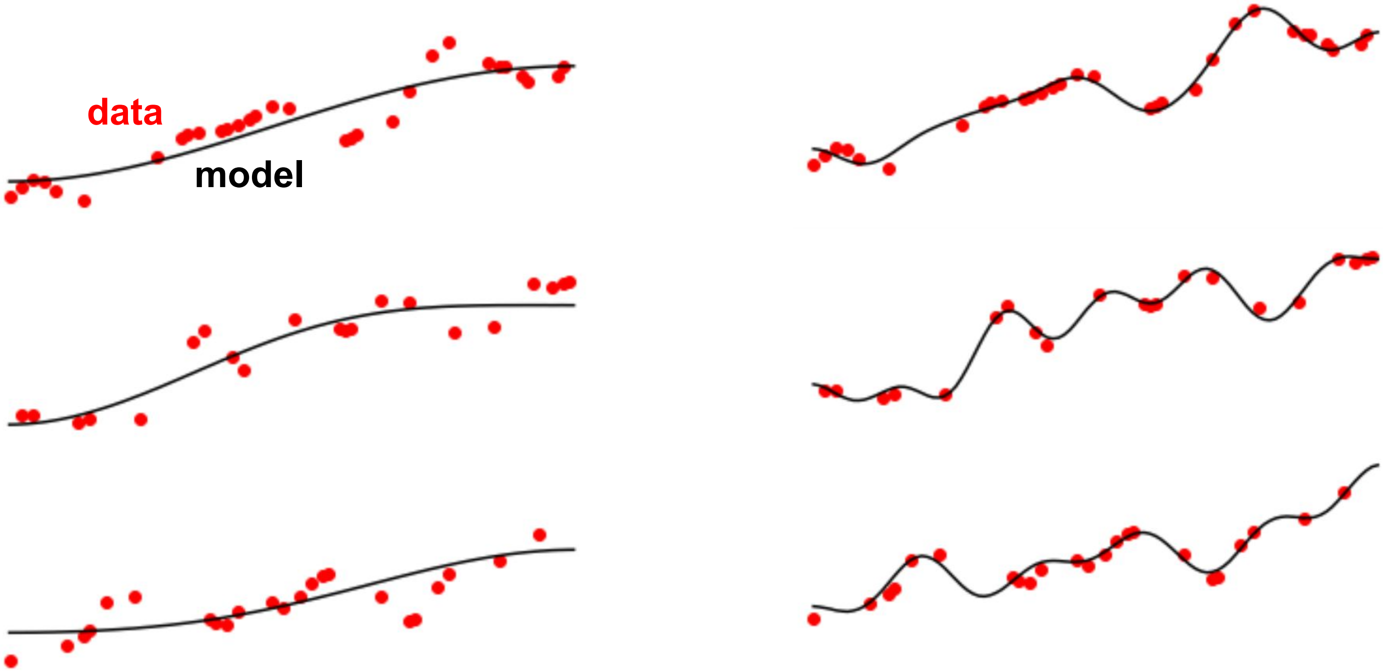
Low-variance, high-bias
(usually fewer parameters)

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

Experiment 1

Experiment 2

Experiment 3



(*but see “double descent”)

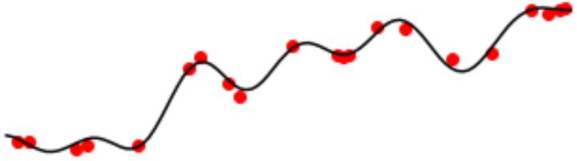
Overfitting

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 ~ # data points ⇒ 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 >> # hyperparameters

Example 1: Linear filter

Filter parameterized by h(0), h(1), … h(T).

Hyperparameter = T (filter length).

Example optimization routine:

For each $T = T_1, \ldots, T_N$: fit h(0:T) to data
Select T with best goodness-of-fit

Example 2: Artificial neural network

Parameters = weights

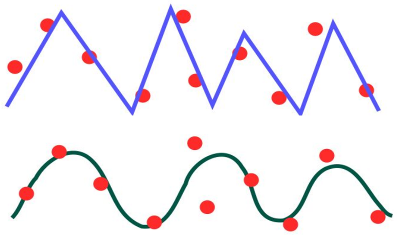
Hyperparameters = # layers, learning rate, …

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)

Multiple models can explain data equally well

Nested model analysis

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

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

If Loss(A’) < 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)?

Random processes perspective (0)

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

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

Distribution over
trajectories x(t)

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

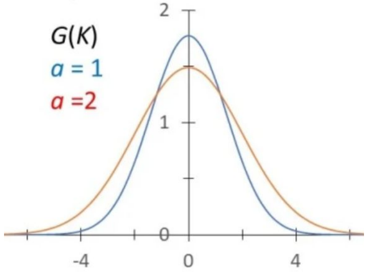
Dataset of *n* sample trajectories

Random processes perspective (I)

Univariate probability distribution

P(x)

*short for P(X=x)

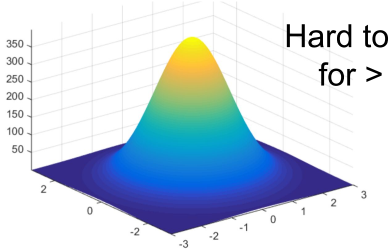


3.13, 2.5, -2.62, 10.23, …

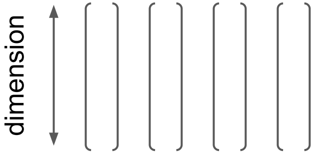
Samples → numbers

Multivariate probability distribution

P(**x**)



Hard to draw for > 2D

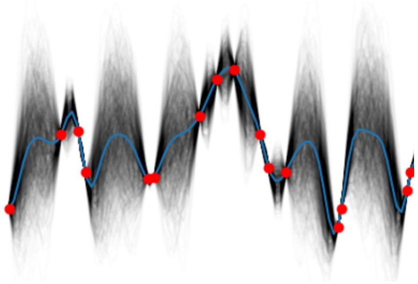


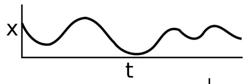
…

Samples → vectors (indexed by i: v_i)

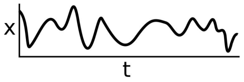
Random process

P[x(t)]





“Ensemble” = infinite collection of samples



…

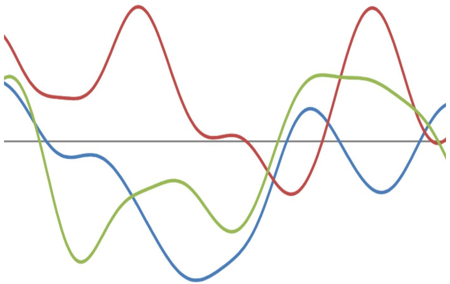
Samples → functions (“indexed” by t: x(t))

Equivalent in code when represented as arrays

Random processes perspective (II)

Common random processes

Gaussian Process

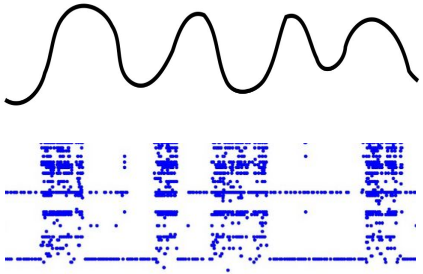


All subsets $[X_1(t), \ldots, X_T(t)]$ jointly Gaussian distributed

Specified by mean and covariance function

Common model of continuous data with structure

Poisson Process



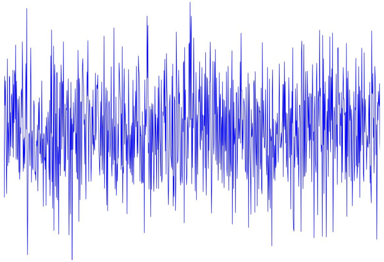
Specified by "rate" $r(t)$

"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

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

Bayes' Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

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

Also Bayes' Rule

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

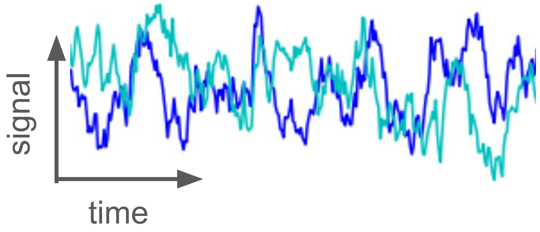
Random processes perspective (IV)

Key random processes concepts: stationarity

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

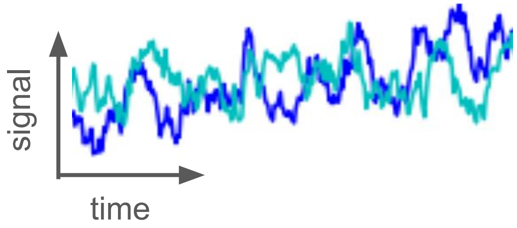
$$P(x(t_1), x(t_1 + \tau)) = P(x(t_2), x(t_2 + \tau))$$

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

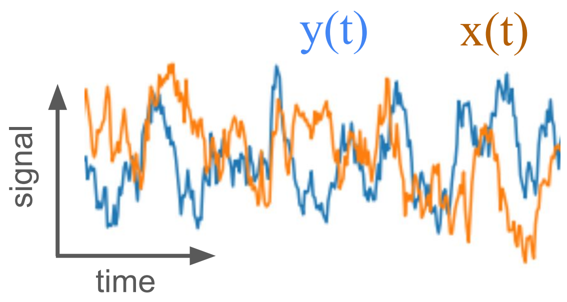
Assumes stationarity
(depends on time lag only)

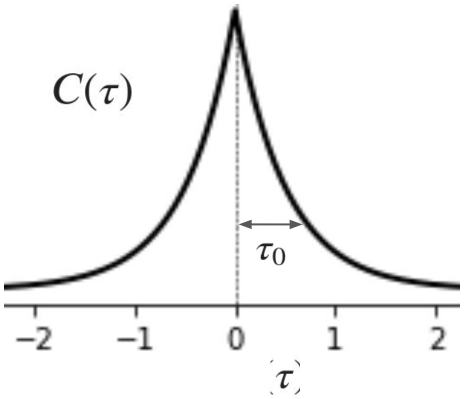
$$C(\tau) = \mathbb{E}[(x(t) - \mathbb{E}[x(t)])(x(t+\tau) - \mathbb{E}[x(t)])]$$

"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

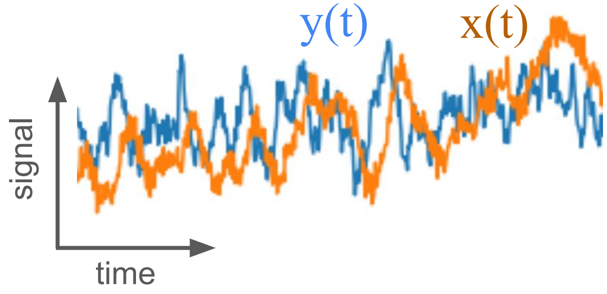
Property of model P[x(t),y(t)] but can estimate from data

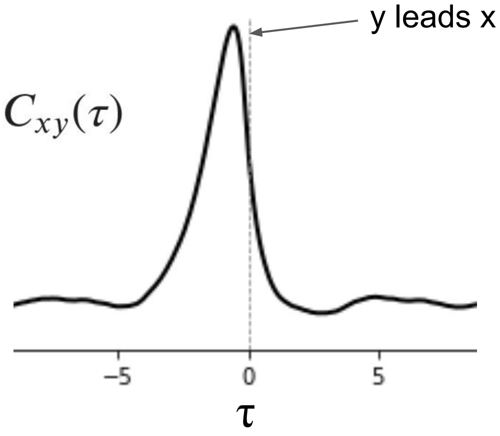
$$C_{xy}(\tau) = \mathbb{E}[(x(t) - \mathbb{E}[x(t)])(y(t+\tau) - \mathbb{E}[y(t)])]$$

Assumes stationarity
(depends on time lag only)

Key statistic describing how two time-series are related

- Useful for detecting leading/lagging processes





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

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

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

Generally not symmetric
(unlike autocovariance)

Random processes perspective (VII)

Comparing/fitting random process models to data

Bayes' Rule

Likelihood Prior

Posterior

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

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.

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

Maximum a posteriori (MAP)

Maximize

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

with respect to θ.

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

Marginal likelihood

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

> Useful for comparing model classes

Random processes perspective (VIII)

Learning and inference

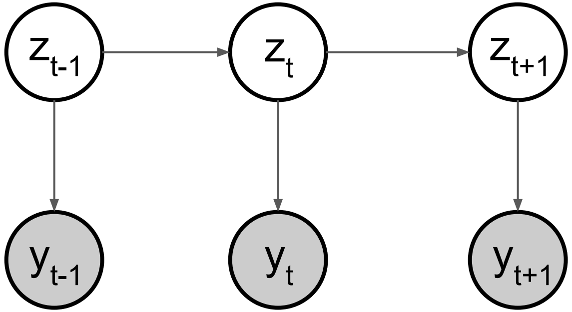
Many models have both “hidden states” + parameters

- 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



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

Dynamical systems perspective (0)

Generative model of data = *equations of motion*

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

x: state　　　　**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: $\mathbb{R}^N$

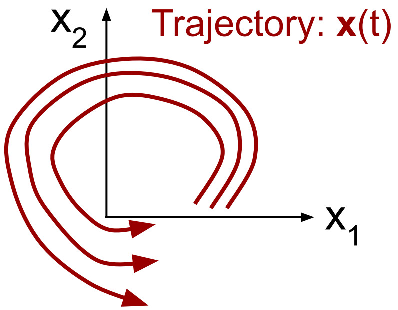
$$\frac{d\mathbf{x}}{dt} = \underline{\mathbf{f}(\mathbf{x})} \longleftrightarrow \frac{dx_i}{dt} = f_i(x_1, \ldots, x_N)$$

vector notation component notation

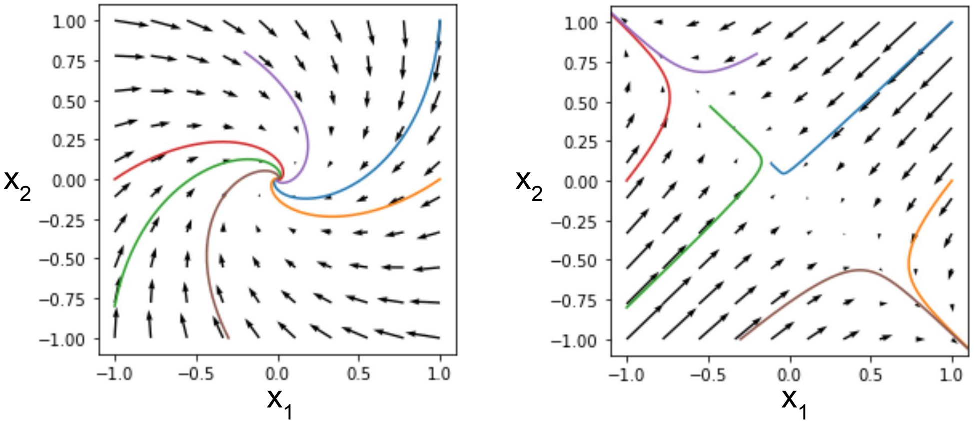
Flow field (also in $\mathbb{R}^N$)

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

Discrete-time version

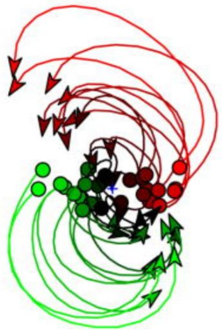


- Rarely "solve" for $\mathbf{x}(t)$
- Typically: characterize fixed points, timescales, limit cycles, sensitivity to initial conditions, etc



Trajectory determined by initial condition $\mathbf{x}(0)$ and flow field

(hard to visualize in > 3D)



Churchland et al 2012
(macaque motor cortex)

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

Dynamical systems perspective (II)

Linear dynamical systems

"Neural" form

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

Specified by single matrix
(+ optional neural timescale τ)

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

Discrete-time version

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

"Driven" version

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

Typically decay to zero or blow up to infinity

- Depends on max eigenvalue of 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 → decay/growth dynamics
Imaginary eigenvalues → rotational dynamics
Complex eigenvalues → decay/growth/rotational dynamics

Dynamical systems perspective (III)

Recurrent neural networks

Rate RNN

"Voltage" / Weights / External inputs

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

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

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

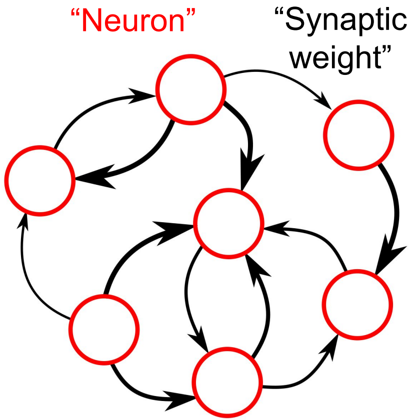
$$= (1 + \tanh(\mathbf{h}))/2$$

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

"Nice" (continuous, differentiable)

"Neurons" can model real neurons

OR used as generic flexible model



Spiking RNN

Spike trains

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

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

Emit spike.

Reset to 0 (or v_{reset}) for τ_{RF}.

Not differentiable

Usually model of real neurons

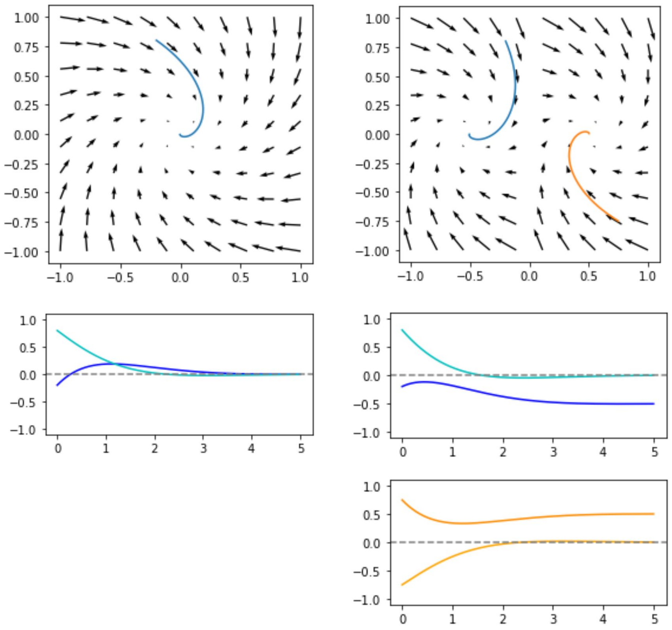
LIF (leaky integrate-and-fire) is most common model

But many variations (e.g. adaptive LIF)

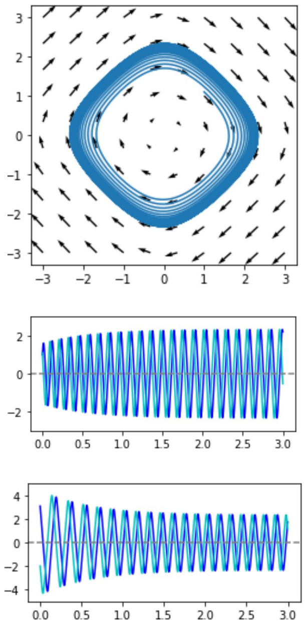
Dynamical systems perspective (IV)

Commonly studied phenomena

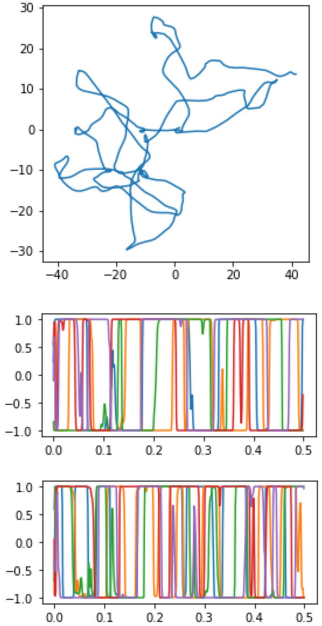
Fixed points



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)

(But don't need to know stochastic calculus for most data analyses)

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

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 **x**(0) and inputs **u**(t) can also be source of randomness

Example 1:
x(t) = firing rates

Example 2:
x(t) = hidden computational variables,
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)