Neuroinformatics
- Linear Time Series Aanlysis (6) -

2020.12.8

Prof: Boreom Lee

Computational and Neurocognitive Time Series Models

- Most of the time series models so far (Sects. 72.-7.5): general purpose models relating consecutive measurements through time in arbitrary time series
  - Some of them – already formulated with neural systems
  - Variables and parameters in these models: do not per se have any meaning but find their specific interpretation in the scientific context at hand
- The statistical machinery for parameter estimation and testing so far – in principle also applied to time series models constructed from variables which per se represent specific theoretical quantities (ex: in models of cognitive or neural processes)
- Or (formulated more from the statistical perspective), incorporating domain-specific knowledge into statistical model inference (from the outset) – boost our ability to detect scientifically meaningful patterns in the data

Computational and Neurocognitive Time Series Models

- A recent trend in theoretical and cognitive neuroscience, especially in the area of reinforcement learning and decision making
  - Explanatory, computational models combined with probability assumptions: a powerful approach to look deeper into the computational mechanisms → generating the data at hand & directly probing assumptions that provide a theoretical explanation
  - Computational models firmly placed into a statistical framework this way → possible to directly test different hypotheses about the computational mechanisms that presumably underlie the observed data
  - Formal criteria (like estimators of prediction error) according to which one can judge the appropriateness of the developed model for explaining the data

Computational and Neurocognitive Time Series Models

- Computational reinforcement learning theory
  - A branch of machine learning originated from findings in behaviorist psychology
  - Centered around the idea that organisms strive to maximize their present and future rewards
  - Each situation-action pair (*s*, *a*)
    - A value (function) $V(s,a)$ – iteratively updated with repeated experience on (s,a) for each situation *s* and each action *a*
    - The simplest case
      - Update rule: $V_{t+1}(s,a) = V_t(s,a) + r_{t+1}$
        - Follows the amount of reward r_t the animal received at time t upon executing a in s
        - Punishment – conceived as negative reward in this framework
        - Linear; a perfect integration & thus nonstationary

Computational and Neurocognitive Time Series Models

- More sophisticated agents (animals)
  - Usually take future rewards into account that follow from choosing a in situation s
  - Future rewards – usually discounted by some factor $\gamma^{\Delta t}$ as a function of time
  - The expected sum of present and future discounted rewards when choosing the optimal path of actions:
    $V_t(s_t, a_t) = \mathrm{E}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \,|s_t, a_t\right]$ (a)
    - The total value of (s, a) reflect this expected sum
    - In each subsequent situation $s_{t+\Delta t}$ that action $a_{t+\Delta t}$ maximizes reward prediction
    - Makes sense from an evolutionary and economical perspective
      - As the future is uncertain (the world is not stationary)
      - As this uncertainty usually grows the more distant into the future an event is (γ may be seen as a way to implement this decay in reward probability)
        - (or) More proximal rewards – seem more valuable partly because the lifetime of an animal is limited

Computational and Neurocognitive Time Series Models

- More sophisticated agents (animals)
  - Value-update rule:

$$\mathrm{E}\left[\sum_{i=0}^{\infty}\gamma^{i}r_{t+i}\right] = \mathrm{E}[r_t] + \gamma\mathrm{E}\left[\sum_{i=1}^{\infty}\gamma^{i-1}r_{t+i}\right]$$
$$\Rightarrow V(s_t, a_t) = \mathrm{E}[r_t] + \gamma V(s_{t+1}, a_{t+1}) \qquad \text{(b)}$$

    - The idea incorporated either way
    - Consistent reward prediction across subsequent situations and actions
    - Assumption
      - The agent chooses the optimal action in each situation [according to the expectancies given by (a)]
      - (given this) deterministic transitions among situations
        - (For clarity) the explicit dependence of the expectance values on s_t, a_t – dropped from the notation
        - Derived by expanding (a) directly & emphasizing here the temporal consistency requirement

Computational and Neurocognitive Time Series Models

- More sophisticated agents (animals)
  - The difference btw the bottom left- and right-hand sides in (b):
    $\partial_t := \gamma V(s_{t+1}, a_{t+1}) + r_t - V(s_t, a_t)$ (c)
    - r_t (observed reward): the temporal difference error (TDE, or reward prediction error)
    - Used to update the values in each step according to
      $V_{t+1}(s, a) = V_t(s, a) + \alpha\partial_t$ (d)
      - α: a learning rate
  - Updating according to future expected rewards
    - Interpreted as a kind of "reward diffusion" from future situations to the present one
    - A process related to the idea of higher-order (secondary, tertiary, etc.) reinforcers

Computational and Neurocognitive Time Series Models

- Neurophysiological findings
  - TDE – related to the activity of neurons in the midbrain ventral tegmentum (VTA) and substantia nigra (SN) → steered a wave of excitement in the animal and artificial learning literature
  - Classical and operant conditioning tasks
    - VTA/SN neurons – initially respond (via firing rate increase) only to the unconditioned stimulus (US)
    - During the course of learning: these responses to the now expected US – vanish and shift to the predicting conditioned stimulus (CS)
    - When a predicted US is omitted: VTA/SN – respond at the expected time of occurrence with a temporary decrease in their firing rate
  - Occurrence and sign of firing rate changes in VTA/SN neurons – appear to comply with (c)

Computational and Neurocognitive Time Series Models

- The basic form of the model above
  - Value V: a function of the present state-action pair (s, a) only
  - Model: 1st-order Markovian like the state space models introduced in Sect. 7.5.1
    - Information about the past – incorporated into the present state representation s (e.g., in the form of a short- or long-term memory trace) → s may encompass internal in addition to external information
    - Easily extended to continuously valued state and action representations
      - For instance, when s represents variables like movement velocity or direction, or a the amount of force applied with the hand
      - V: in this case, not defined in terms of a lookup table $(s, a) \rightarrow V(s, a)$ as in the discrete state space case, but as a continuous (regression) function s and a

Computational and Neurocognitive Time Series Models

- How values $V(s,a)$ are translated into choices of a particular path of actions?
  - A common choice: a "Boltzmann-type" decision function

$$pr(a_t = k | s_t = m) = \frac{e^{\beta V(k,m)}}{\sum_l e^{\beta V(l,m)}} \qquad \text{(e)}$$

    - An action a chosen in situation s with a probability corresponding to that action's value
  - Parameter β: an "inverse temperature" which regulates the "flatness" of the decision landscape
    - $\beta \to \infty$: the agent will deterministically always choose the highest-valued action ("exploit")
    - $\beta \to 0$: it will choose each action with equal likelihood regardless of value ("explore")
    - ➢ Determine the "exploitation-exploration tradeoff"

Computational and Neurocognitive Time Series Models

- Computer science and artificial intelligence
  - A burgeoning literature on how to use such learning algorithms and derivatives thereof to train artificial agents (robots) to perform tasks like playing backgammon or checkers, balancing, planning and heuristic search, and so on

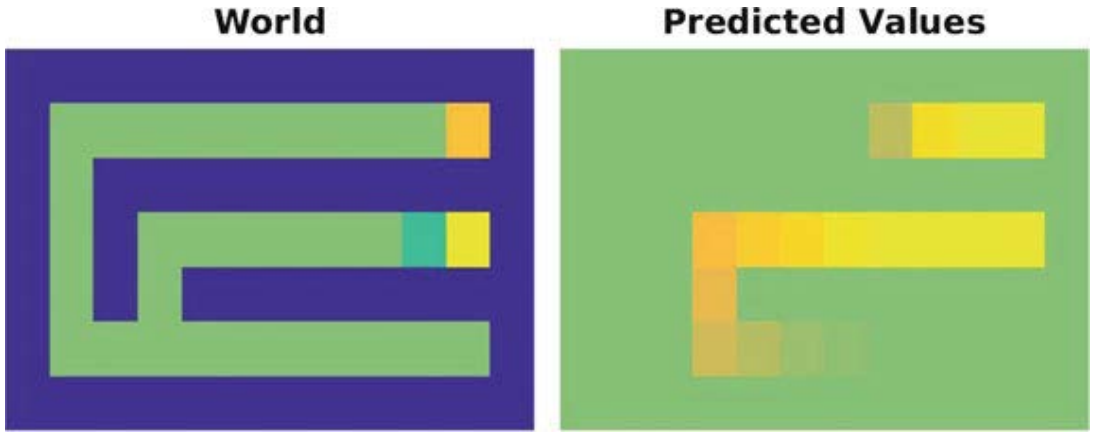

Fig. 7.9 Illustration of reinforcement learning on a virtual maze. Left graph shows the maze with reward locations and magnitudes (reddish and yellow squares) and current position of the agent (dark green), as implemented in MATL7_8. Right matrix displays estimated discounted future reward values after several runs

Computational and Neurocognitive Time Series Models

- Parameter estimation & model fitting to experimental data to gain some insight into underlying mechanisms
  - A likelihood function for the system parameter $\boldsymbol{\theta} = \{\beta, \alpha, \gamma, V_1\}$:
    $p(\{(s_t, a_t)\}|\alpha, \beta, \gamma, V_1) = \prod_{t=1}^{T} p(s_t, a_t|\{s_\tau, a_\tau|\tau < t\}, \boldsymbol{\theta}) =$
    $\prod_{t=1}^{T} p(a_t|s_t, \{s_\tau, a_\tau|\tau < t\}, \boldsymbol{\theta}) p(s_t|\{s_\tau, a_\tau|\tau < t\}, \boldsymbol{\theta})$ (f)
    - Given a time series $\{(s_t, a_t)\}$ $(t = 1, \ldots, T)$ of experimentally observed situations and actions performed by an animal
    - V_1: the set of initial values
- TDE-model (as specified above)
  - Interpreted as a generalized state space model with deterministic linear transition equation (c)-(d)
  - Nonlinear measurement equation (e)
    - Links the outputs a to the underlying state values V by means of the multinomial distribution in the discrete case → the linearity in the transition: the reason why included this model class here, in Chap 7, rather than with Chap. 8 or 9
  - The update equations for the hidden state V – deterministic in the basic model (the sequence of "innovations" r_t – observed as well) → simplifies estimation considerably compared to a full state space model (∵ one dose not have to integrate across sets of unobserved state paths)

Computational and Neurocognitive Time Series Models

- TDE-model (as specified above)
  - Nonetheless, scientifically it may be more appropriate to account for randomness in the value updating as well
    - Inference for the deterministic transitions – otherwise derailed by noise fluctuations in the true generating process (as noted in Sect. 7.5.1)
  - A full state space framework
    - Offer some protection against errors introduced by invalid model assumptions
    - Approximate EM schemes – available for this case with non-Gaussian observations (as discussed in Sects. 7.5.3 and 9.3.1)
- Fixed effects qua experimental design
  - Scenario where transitions among states s and reward feedback r are deterministic given the actions a
    - i.e., $p(s_t|\{s_\tau, a_\tau|\tau < t\}) = 1$ for one specific situation m and 0 for the others
  - Everything is fully determined by the course of actions taken & all terms depending solely on s could be dropped from (f)
  - All knowledge about the past course of actions – embodied within the current values $V(s, a)$

Computational and Neurocognitive Time Series Models

- Fixed effects qua experimental design
  - The log-likelihood simplified to: $\log p(\{a_t\}|\boldsymbol{\theta}) = \sum_{t=1}^{T} \log p(a_t|s_t, \{s_\tau, a_\tau|\tau < t\}, \boldsymbol{\theta}) = \sum_{t=1}^{T} \log \frac{e^{\beta V_t(a_t=i|s_t,\alpha,\gamma)}}{\sum_j e^{\beta V_t(a_t=j|s_t,\alpha,\gamma)}} = \sum_{t=1}^{T} \left[\beta V_t(a_t = i|s_t, \alpha, \gamma) - \log \sum_j e^{\beta V_t(a_t=j|s_t,\alpha,\gamma)}\right]$ (g)
    - i: the actually chosen (by the experimental subject) action on each trial
    - The transition process for V_t (and the environment) – deterministic → V_t – completely specified by the actual path of actions $\{a_t\}$ taken by the animal and the actual reward feedbacks received (i.e., spelled out as explicit functions of parameters α, γ, and V_1)
    - Then proceeds as usual by maximizing (g) w.r.t. β, α, γ, and V_1
- Formal statistical tests
  - $H_0: \gamma = 0$ (only present, no future rewards considered)
  - Conducted using likelihood ratio statistics as long as models are strictly nested
  - (otherwise) criteria like AIC, BIC, or CVE-based procedures used
    - For model comparison and selection
    - Or to assess the prediction quality of a given model in the first place (Chap. 4)
  - Uncertainty in the environment or noise in the value updating → a more elaborated state space approach have to be considered (Sects. 7.5 and 9.3.1)

Computational and Neurocognitive Time Series Models

- RFL models used increasingly in the analysis of behavioral, human neuroimaging or animal in vivo electrophysiological data
  - Often first estimated by maximum likelihood or Bayesian inference from the observed behavioral data
  - Model parameters (e.g., learning rate α or discount factor γ) or variables (e.g., $V(s,a)$) – subsequently used as predictors in (usually linear) regression models to account for variation in the simultaneously recorded neural activity
  - Khamassi et al. (2014)
    - Used RFL for the analysis of in vivo electrophysiological recordings by first probing which of a variety of RFL model variants could best account for the observed behavioral data in a sequential search task (using criteria like the BIC, Sect. 4.1)
    - Best-performing model – then used to analyze single-neuron recordings from the lateral prefrontal cortex and dorsal anterior cingulate cortex to figure out the neural processes underlying the behavioral performance
      - In particular, the distinct roles of these two brain areas and mechanisms which trade exploration initially in trials for exploitation later on [i.e., the regulation of β in (e)]

Computational and Neurocognitive Time Series Models

- RFL models used increasingly in the analysis of behavioral, human neuroimaging or animal in vivo electrophysiological data
  - Sul et al. (2010)
    - Used FRL models to examine how processes like prediction errors or value updating were neuronally represented in in vivo electrophysiological recordings from the rodent orbitofrontal and medial prefrontal cortex
  - Koppe et al. (2017)
    - RFL models – inferred from data to gain insight into behavioral processes per se,
    - E.g., for determining the kind of behavioral strategy employed by animals in solving a task
- Approximate numerical techniques or sampling schemes
  - To evaluate likelihood functions or posteriors (cf. Sect. 9.3)
  - Used in the more complicated, nonlinear cases, to estimate any formal model of an animal's behavior from observed data
    - RFL models – probably the ones employed most frequently with this type of approach, not the only ones which could be used

Computational and Neurocognitive Time Series Models

- Approximate numerical techniques or sampling schemes
  - Belief learning models (Camerer and Ho 1999)
    - A close relative of RFL models
    - Originated in economics to account for subjects' learning behavior in game-theoretical setting
    - Similar to RFL models in the sense that they contain update mechanisms upon experienced outcomes (returns)
    - RFL – encompassed as a special case for specific settings of the parameters, at least in the formulation by Camerer and Ho (1999)
    - Parameters obtained by maximum likelihood principles from observed experimental data (like for RFL models)
    - The beliefs about what the subject would have earned had it chosen a different action given the opponent's response – updates as well
      - In conventional RFL models: only values for the selected action and current situation – updated
    - Ex: used in the analysis of how genetic differences (exploiting natural polymorphisms) affects learning in strategic settings (Set et al. 2014)

Computational and Neurocognitive Time Series Models

- Approximate numerical techniques or sampling schemes
  - Noisy accumulator models (Brunton et al. 2013)
    - Based on drift-diffusion models introduced by Ratcliff and McKoon (2008)
    - For the process of evidence integration and decision making in rats and humans
    - The form of a linear state space model with Gaussian noise and sensory inputs
      - A "decision variable" – triggers the subject's choice once a threshold is crossed
      - A variable – models sensory adaptation
    - Parameters – estimated from the series of subjects' choices by maximum likelihood
    - Ex: used to differentiate the role of various internal and external noise sources in the decision-making process
- Final remark
  - Most of these behavioral models – linear in their transitions (included here in Chap. 7)
  - Linear models
    - Quite limited in the repertoire of dynamical phenomena (as clear in Chap. 9)
    - Cannot produce stable oscillations on their own
  - A shift to nonlinear modes – required for a number of interesting behavioral time series

Bootstrapping Time Series

- Time series data: potentially a highly complicated dependency structure and unusual distributional properties
  - Dependent on the precise nature of the generating dynamical system (Chap. 9) and the level of noise
  - Ex: the membrane potential distribution produced by a spiking neuron
- Physiological distributions
  - Quite sharp cutoffs, unusual trail, or multimodality induced by biophysical processes and constraints
  - Ex: the absolute refractory period limiting the maximum spike rate or reversal potentials confining the voltage distribution

Bootstrapping Time Series

- Several test statistics for linear and generalized models based conventional parametric distributions
  - Based on the idea that we can neatly separate a systematic time-varying part from a purely (usually Gaussian) white noise process with no temporal dependencies
  - In principle, possible even in more complex, nonlinear cases, e.g., if a good process model of the spiking behavior in the example above
  - Often not at all trivial
    - With time series, much more cautious in applying conventional parametric assumptions than with iid data
    - Parametric time series tests – usually require properties like stationarity and ergodicity not that easy to establish in practice
  - Often only one time series, not a sample of several series produced under identical conditions & sometimes even quite short series (always problematic for asymptotic statistics)
    - Ex: time series from fMRI
    - Parametric significance levels obtained in the time series context – sometimes only taken as a guidance rather than for strict hypothesis testing

Bootstrapping Time Series

- One particular complication for the methods in this chapter
  - In the real world & in the brain (in particular): the processes underlying the observed time series – rarely linear (see Chap. 8)
  - Linear models
    - Still a good approximation, especially if the noise is large and the linear part dominates the dynamics
    - Still provide useful information about salient features of the series
  - Also possible to remove strongly nonlinear features (e.g., cutting out spikes) or linear models just serve as null hypothesis reference → situations where still going ahead with linear models although the residuals may violate Gaussian white noise assumptions to some degree

Bootstrapping Time Series

- Bootstrap and permutation methods
  - An alternative to circumvent some of the problems of parametric test for checking significance
  - Recommendation: to back up parametric inferences drawn from time series by bootstrapping methods
  - Both parametric as well as nonparametric forms of the bootstrap
- Parametric bootstrap or permutation test
  - Usually based on the residuals from a fitted model
  - AR(p) model: $x_t = a_0 + \sum_{i=1}^{p} a_i x_{t-i} + \varepsilon_t, \varepsilon_t \sim \mathrm{N}(0, \sigma^2)$ (h)
    - Parameter estimates $\{a_i\}$ – obtained as outlined in Sect. 7.2.1
    - Variance σ^2 – estimated from the residuals: $\hat{\sigma}^2 = \frac{1}{T-2p-1} \sum_{t=p+1}^{T} \hat{\varepsilon}_t^2$
      - $T - p$: the length of time series available
      - Parametric bootstrap samples $\{x_t^*\}$ – created by randomly drawing numbers $\varepsilon_t^* \sim \mathrm{N}(0, \hat{\sigma}^2)$ & process (h) – iterated forward in time based on these
  - Start with initial estimates for the first p observations $\{x_1^*, \ldots, x_p^*\}$ & then iteratively update x_t according to (h)
  - AR(p) model (h) – then refitted on each of these bootstrap samples to obtain SE estimates for the parameters $\{a_i\}$ or a reduced model – fitted using the estimated residuals for a formal hypothesis test

Bootstrapping Time Series

- Parametric bootstrap or permutation test
  - A crucial point different from parametric bootstrapping in linear regression = to iterate the process (h) through time
    - Not just randomly resample $\varepsilon_t^* \sim \mathrm{N}(0, \hat{\sigma}^2)$ & add to the extracted systematic part $\hat{x}_t$
    - ∵ This destroy the temporal consistency of the model, i.e., the requirement $\varepsilon_t = x_t - \left(a_0 + \sum_{i=1}^{p} a_i x_{t-i}\right)$ in model (h) → the bootstrap process would not follow anymore the estimated AR(p) model that underlie the observed data
  - Dropping Gaussian assumption together
    - Otherwise only little advantage over parametric tests as offered in Sect. 7.2.2
      - In the parametric bootstrap setting: make the residuals conform to a Gaussian
      - With the asymptotic test: they are to begin with
    - Assuming $\mu_\varepsilon = 0$: start by centering the residuals $\varepsilon' = \varepsilon - \mathrm{avg}(\varepsilon)$
    - Retaining the white noise idea, i.e., the independence of residuals $\varepsilon_t (E[\varepsilon_{t'} \varepsilon_t] = 0$ for $t' \neq t$): either just create B random permutations of $\boldsymbol{\varepsilon} = \{\varepsilon'_0, \ldots, \varepsilon'_T\}$ or draw T values from $\boldsymbol{\varepsilon}$ with replacement as in the classical bootstrap
    - Based on these, process (h) – iterated in time as described above

Bootstrapping Time Series

- Linear model for simplicity or convenience
  - From inspecting the residuals, a linear model does not capture all the dependencies in the series
  - The assumption $E[\varepsilon_{t'}\varepsilon_t] = 0$ – violated to some degree
- Other bootstrap/permutation strategies specific for time series
  - Block permutation or block bootstrap
    - The whole time series of length T into K blocks of size M ($T \leq K \times M$)
    - Instead of permuting or bootstrapping individual ε_t, permute or draw from whole blocks of M temporally consecutive ε_t values
      - Randomly rearrange K non-overlapping sets $\{\varepsilon'_t, \ldots, \varepsilon'_{t+M-1}\}$ into a new time series or
      - With bootstrapping, draw K sets from these with replacement and concatenate them
    - Idea here
      - These bootstraps retain the temporal dependency structure of the original series
      - The block length M – chosen large enough so that any interdependencies (or autocorrelations) have largely died out after steps M
      - Breaks will remain across the block edges

Bootstrapping Time Series

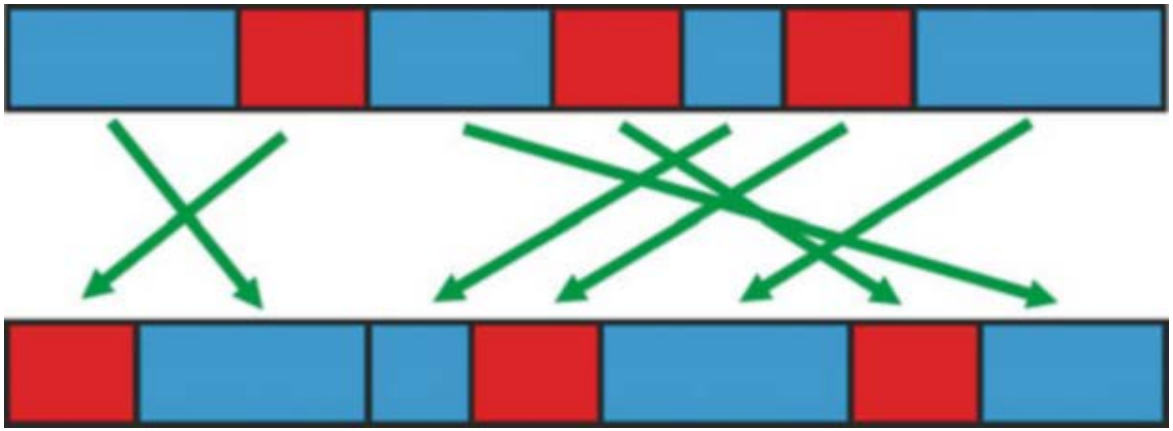

Fig. 7.10 With block permutation bootstraps, whole blocks of consecutive values from the original time series (top) are randomly shuffled (bottom). If a hypothesis about properties of a time series in relation to experimenter-defined class labels is to be tested, e.g., as with neural recordings in a behavioral task with different stages (blue and red task phases in the graph), one simple strategy is to shuffle blocks of consecutive class labels while leaving the original neural time series completely intact. Reprinted from Durstewitz and Balaguer-Ballester (2010) with permission

Bootstrapping Time Series

- Other bootstrap/permutation strategies specific for time series
  - Block permutation or block bootstrap
    - Select a proper M by inspecting the autocorrelation and/or auto-mutual-information function of the series & cut off when this falls below a certain threshold (e.g., when no deviation from zero anymore)
    - The # of blocks K: should be large enough to allow for a sufficiently large # of distinct permutation or bootstrap samples
      - Permutation case: a total of possibilities $K!$ to arrange the blocks
      - Bootstrapping: K^K – should not be too small (perhaps >1000 if possible, as a rough guideline) to avoid too large of a variance in the estimated p-value
    - For the completely nonparametric case
      - No longer bound to any model assumptions with block permutations/bootstrap
      - Simply dissect the original series $\{x_t\}$ (rather than the residuals) into K blocks & shuffle them around or draw from them with replacement
      - The continuity of the original time series – broken at the $K-1$ interim block edges → several strategies

Bootstrapping Time Series

- Other bootstrap/permutation strategies specific for time series
  - Phase randomization
    - Another popular bootstrapping idea for time series
    - Based on the equivalence of power spectrum and autocorrelation function of a time series
    - Idea:
      - To compute the Fourier transform (power spectrum) of the time series, scramble the phases associate with each frequency component, and transfer back to the time domain
      - Wiener-Khinchin theorem: this also preserve the original autocorrelation function
    - These bootstrap data – consistent with any stationary ARMA model (generating the data)
      - Up to the limitations imposed by the finite length & sampling rate of the observed process
      - A stationary ARMA process – completely specified by the autocorrelation function through the Yule-Walker equations & a Gaussian noise process – completely specified by moments up to 2^{nd} order as well
      - Do not even have to know or estimate the underlying ARMA model; phase randomization preserve the linear-Gaussian model whatever it is

Bootstrapping Time Series

- Other bootstrap/permutation strategies specific for time series
  - Phase randomization
    - Retains only the 1st and 2nd moments of a time series (means and auto-covariances) unlike block permutation
      - Fully specify any stationary Gaussian process but will destroy any nonlinear dependency (higher moments) in the time series
      - This property – used to check for nonlinear structure in time series
  - Applications in neuroscience
    - Block permutations to control for autocorrelations (or any higher-order dependencies)
      - Used widely in various situations with in vivo electrophysiological recordings
      - H_0 distribution of the test statistic: difficult to determine
    - The shuffling of whole (identical) trials (Trial permutation bootstraps)
      - One specific, commonly employed form of block permutation bootstraps in neuroscience
      - To probe whether temporal relations among neurons bear information beyond the single-unit activities considered independently
        - For each recorded unit i, a set of time series $\{x_{it}\}^{(k)}$ (one for each distinct trial k)→ the assignments $\{x_{it}\} \rightarrow k$ – randomized
        - Done for each unit independently → bootstrap data sets $\{\mathbf{x}_t\}^{(k^*)}$ which presumably preserve the trial-specific autocorrelative structure and potential rate variations for each unit, but destroy the interrelations among them

Bootstrapping Time Series

- Other bootstrap/permutation strategies specific for time series
  - Applications in neuroscience
    - The shuffling of whole (identical) trials (Trial permutation bootstraps)
      - Implicit assumptions
        - The observed set of trials – stationary → the different trials – identical in terms of single-unit behavior [(7.7) and (7.8)]
          - If not the case (i.e., if rate variations across trials common to all neurons), these – destroyed as well in the bootstrap data & the inference drawn from them – no longer valid → Semi-parametric bootstraps constructed to avoid this
    - Semi-parametric bootstraps
      - For each neuron, first a kernel density estimate of the spike density (instantaneous rate) performed (using the methods from Sect. 5.1.2) and then spike trains redrawn at random from the estimated density
      - Preserve the rate variations and covariations across trials and neurons; but destroy any finer temporal structure and relationships if present
      - "spike dithering": a similar technique

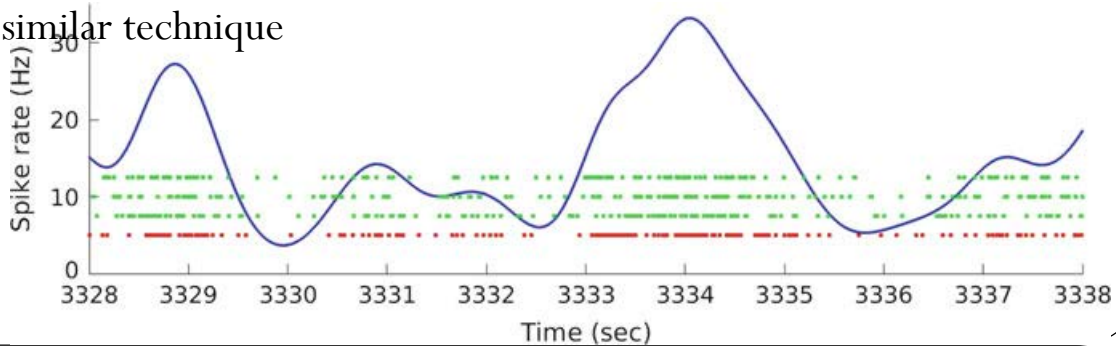

Fig. 7.11 Semi-parametric bootstraps (green dots) from a point process (red dots) preserving the original rate fluctuations as estimated through KDE (blue curve) MATL7_9

Bootstrapping Time Series

- Other bootstrap/permutation strategies specific for time series
  - Applications in neuroscience
    - Trial permutation bootstraps
      - Also employed to probe for coding of stimulus information in the neural activity
        - When trials not identical but grouped according to the experimentally enforced stimulus conditions
      - May simply shuffle the class labels (the assignments $\{\mathbf{x}_t\} \rightarrow C$, Fig. 7.10)
        - If the interest is only in whether the recorded ensemble responses contain significant information about the stimulus
      - May shuffle single-unit assignments within identical trials from the same class C in addition to probe
        - Whether the temporal relations among units contribute to stimulus decoding, or
        - Whether the set of single-unit activities considered independently is sufficient

Bootstrapping Time Series

- Other bootstrap/permutation strategies specific for time series
  - Applications in neuroscience
    - Trial permutation bootstraps
      - (More generally) could shuffle consecutive blocks of class labels (blocks of consecutive time bins belonging to the same event class)
        - Even if not dealing with a set of discrete trials but rather continuous recordings during an extended task with different stimulus and behavioral events
        - To examine whether neural activity discriminates among the different task events (Fig. 7.10)
      - Leave the temporal structure within the neural recordings completely but rather just scramble its relation to different task phases
    - Phase-randomized bootstraps
      - Also various applications in neuroscience
      - For testing whether neural time series significantly deviate from linear dynamic model assumptions
        - E.g., whether they harbor predictability that only complies with a nonlinear process (Sect. 8.1)

Bootstrapping Time Series

- Proper bootstrapping of time series
  - A highly important issue in neuroscience (Mokeichev et al. 2007)
  - Recurring motifs in in vivo recorded membrane potential traces (i.e., segments of V_m traces appearing to repeat with high similarity)
  - Some stunning examples of such repeats, sometimes even minutes apart
    - Confirm underlying microcircuits with strong intra-circuit connectivity that generate highly reproducible membrane potential trajectories once triggered
    - (i.e.) Repeating membrane potential segments – taken as signatures of a fixed sequence of synaptic interactions and thus cell spikings
    - Surprisingly, completely reproduced in various forms of time series bootstraps, based on block permutations or model-generated voltage traces → building blocks of a neural language – may really be d/t unspecific autocorrelative and deterministic structure generated by the membrane potential dynamics