Probabilistic models for neural data
Session 8: State space models #3

To do before this session:

- **Read Yu et al. (2009) and prepare assigned presentation**
- **Read up on this session's statistical concepts**
- **Complete pre-session quiz**

In this session, we will start with discussing Yu et al. (2009), which assumes that the time-evolution of the low-dimensional latent states that generate high-dimensional neural activity is described by a Gaussian process. After that, we will move on to the use of flexible artificial neural networks for models of the latent state's time-evolution. In particular, next session's paper uses variational autoencoders that combine variational Bayesian methods with artificial neural networks, both of which are beyond the scope of this course. Fortunately, understanding their exact functioning won't be required to get the main ideas of the next session's paper. For this reason, I provide a brief outline of variational autoencoders below, and during this week's session.

Paper: Yu et al. (2009). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity.

The paper infers single-trial latent dynamics from neural activity by assuming Gaussian square-root neural activity and latent dynamics described by a Gaussian Process. It was one of the first widely accepted neural state-space models and the described method is still in frequent use nowadays. When reading the paper, please focus on the following:

- Why is it beneficial to investigate single-trial neural dynamics rather than neural activity averaged across trials of the same condition?
- What are the benefits of FA over PCA for dimensionality reduction of neural spike counts? Why does PPCA outperform PCA?
- What are the benefits of GPFA over the two-stage methods discussed in the paper?
- Why do they use the square-root transform and a Gaussian noise model rather than a Poisson noise model?
- Why do they measure the predictive quality of a model by their leave-neuron-out rather than alternative, probabilistic methods, like the cross-validated data likelihood?
- What are the benefits of the additional orthonormalization step when visualizing GPFA latent state trajectories?
- What does a different characteristic timescale for different latent dimensions imply?
- What is the difference between standard GPFA and reduced GPFA?
- Why does FA even with optimized parameters not match the predictive performance of GPFA?

Please make sure to read the Methods section, as it contains additional discussion on the benefits of GPFA, and justifies some of the modeling choices. The Appendix contains a few interesting additional facts about GPFA, model fitting, etc., and should be skimmed, while skipping the mathematical details.

Presentations:

1. The benefit of single-trial analysis (Introduction & Fig. 1)
    a. Use the examples in Fig. 1 to describe which details of neural activity might be missed when averaging this activity across trials, and
    b. to describe how combining the activity of multiple simultaneously recorded neurons yields better insight into neural dynamics.
2. Two-stage methods for single-trial spike train analysis (Figs. 2, 3A, 4 & Text/Methods)
    a. Describe the two-stage method for extracting neural trajectories (Fig. 2 & Text/Methods).
    b. Why do the authors consider the use of (P)PCA for dimensionality reduction of neural data problematic, and suggest the use of FA instead (Fig. 4 & Methods)?
    c. Describe why the authors suggest applying FA to square root-transformed spike counts. What problem does the square root-transform solve (Methods)?
    d. Explain the four disadvantages of the two-stage method that the authors mention in the text.
3. Gaussian Process Factor Analysis (GPFA; Methods & Fig. 3B)
    a. Describe the components of GPFA. How do they assume neural activity to be generated from the latent state (emission model; Eq. (1))?
    b. How do they assume the latent state to evolve over time (transition model; Eq. (2))? What are the parameters of the GP kernel (Eq. (3)), and what do they control?
    c. Given neural activity data and fixed GPFA parameters, how are the latent states estimated (i.e., the E-step in the EM algorithm)?
    d. What are the GPFA parameters, and how are they fitted (i.e., the M-step in the EM algorithm)?
    e. How does GPFA solve the four problems that plague comparable two-stage methods (Introduction & Fig. 3B).
4. GPFA and Orthonormalized GPFA applied to data from delayed-reach task (Methods & Figs. 6 & 7)
    a. Briefly describe the delayed-reach task and the resulting neural data (Methods).
    b. Describe the neural trajectories shown in Fig. 6. Why can each latent dimension only have a single time-scale? What is the issue with the ‘raw’ latent dimensions that is solved by orthonormalization?
    c. Qualitatively describe the orthonormalization procedure (Methods).
    d. Describe the orthonormalized neural trajectories shown in Fig. 7. What property of these neural trajectories can be linked to the variance explained in PCA? How does each orthonormalized latent dimension support multiple time scales?
5. Reduced GPFA, dynamic AR(1) models, and performance comparison (Fig. 5)
    a. Describe reduced GPFA as a way to use *p* timescales with a *p*’-dimensional (*p*’<*p*) latent state (Methods).

   b. Describe how GPs can be used to model linear latent dynamics (Methods; Eqs. (4) & (5)). (Advanced) How does the resulting transition model differ from that of Macke et al. (2015)? (Hint: think about the implications of independent latent state dimensions)
   c. Describe and interpret the panels of Fig. 5. How do they compute the prediction error (Method)? Why do you think the error of reduced GPFA is lower than that of GPFA?
6. Reduction of variance (Fig. 8) and effect of delay period duration on reaction time (Fig. 9)
   a. Describe how the authors replicate previous findings of reduced across-trial variability of neural activity when moving from target onset to go cue (Fig. 1D & Fig. 8). Why did they use the 'full' latent space for this analysis, rather than only the first three dimensions shown in Fig. 8? Why can they directly link latent space and neural activity variability?
   b. How do the authors use the neural trajectories shown in Fig. 9 to reason about the relationship between the delay period duration and reaction time?

Statistical concepts: variational autoencoders

Autoencoders are models that have an encoder and a decoder. The encoder encodes some observables *x* to a lower-dimensional latent representation *z*, as can be described by the conditional probability *p*(*z*|*x*). The decoder, *p*(*x*|*z*), recovers this observable from the lower-dimensional latent representation. This makes autoencoders a special case of dimensionality reduction techniques. By Bayes' rule, $p(z|x) \propto p(x|z)p(z)$, linking the encoder and the decoder. Furthermore, $p(x) = \sum_z p(x|z)p(z)$, such that autoencoders provide an (unsupervised) model *p*(*x*) for the observations.

What is special about autoencoders is that encoder and decoder aren't necessarily coupled, as would be required by Bayes' rule. Instead, they might have a separate set of parameters, say $p(z|x;\theta)$ and $p(x|z;\phi)$, that are jointly tuned to minimize the reconstruction error, which is the difference between $p(x)$ and its approximation that results from encoding *x* in *z*, and then decoding it to recover *x* again.

In variational autoencoders, as introduced by Kingma et al. (2013) and Rezende et al. (2013), both the encoder and the decoder are modeled by a feed-forward artificial neural network (ANN), which are trained jointly by the use of variational Bayesian inference. The best way to think about ANNs in this context is as complex function approximators that (in case of the encoder) take input *x* and output the parameters of *p*(*z*|*x*). In case of the variational autoencoder, *p*(*z*|*x*) is assumed to be a multivariate Gaussian, such that the ANN with parameters θ outputs the mean and (co)variance of this Gaussian. The decoder also uses an ANN with parameters ϕ to map a latent state *z* to the parameters of *p*(*x*|*z*), whose form (and therefore the meaning of the parameters) depends on the nature of the observations *x*. In the discussed paper, these observations are continuous variables (i.e., the latent factors) that are, for each time step, modeled by Gaussians, such that the ANN outputs the Gaussians' mean and (co)variance. Variational autoencoders are called *variational*, as they use the variational Bayesian approximation methods (similar to the EM algorithm) to adjust their parameters, θ and ϕ, in order to synchronize encoder and decoder.

Usually, variational autoencoders model individual observations rather than sequences of observations. The paper for next session (Pandarinath et al. (2018)) uses a variant, known as *sequential variational autoencoder*, that uses similar concepts to the ones described above to model whole sequences of observations. It does so by using recurrent ANNs rather than feed-forward ANNs. The main difference between feed-forward and recurrent ANNs is that recurrent ANNs have an internal state that is updated with every additional input, and can as such retain a history of past inputs (and outputs) in its state. This also allows us to generate sequences of outputs that depend on each other. In that sense they are similar to state space models: they maintain a latent state that is updated at each step and generates an (observable) output. As the latent state is all that needs to be known to predict future outputs, they implicitly embed the Markovian assumption. Training recurrent ANNs consists of adjusting their parameters such that they best match observed sequences. This includes adjusting their latent state representation, how they change across time, and how they relate to the provided observations. In sequential variational autoencoders, the encoder $p(z|x_{1:T})$ determines the initial latent state, and the decoder $p(x_{1:T}|z)$ generates the sequence of observations $x_{1:T}$ from that initial latent state.

For more information on variational autoencoders, either read Kingma et al. (2013) and Rezende et al. (2013), or consult one of the many more gentle introductions available online, such as
https://jaan.io/what-is-variational-autoencoder-vae-tutorial/
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Exercise
(to be completed *after* Session 8)
INSTRUCTIONS WILL BE ADDED BEFORE SESSION 8