The Neural Code

Summer term 2022, Christian Leibold, Yuk-Hoi Yiu

Problem Set 5 **—STC/GLM—** **31.05.2022**

23.06.2022

---

1. (STC) (Python exercise) Consider the two filters:

$$a(t) = \exp\left(-\left(\frac{t - 10\text{ ms}}{\sqrt{10}\text{ ms}}\right)^2\right)\cos\left(2\pi\frac{(t - 10\text{ ms})}{10\text{ ms}}\right)$$
$$b(t) = \exp\left(-\left(\frac{t - 10\text{ ms}}{\sqrt{10}\text{ ms}}\right)^2\right)\cos\left(2\pi\frac{(t - 10\text{ ms})}{10\text{ ms}} + \pi/2\right)$$

(a) Create the filters from 0ms to 20ms with a bin size of 1ms. Also, normalize the filter amplitudes by `numpy.linalg.norm`.

(b) Generate a white noise stimulus $x(t)$ with mean 0 and variance 1 for 100s with a bin size of 1ms.

(c) Consider a neural system which takes $x(t)$ as input and produce an output $r(t)$ with a squaring nonlinearity by

$$r(t) = 2\big[x * a\big](t)^2 + 1.5\big[x * b\big](t)^2$$

Compute $r(t)$. To produce a better result, pad an array of zeros in front of the stimulus $x(t)$ before convolving with the filters with `numpy.convolve(..., mode='valid')`. The zero-padding array should have the size of $(\mathrm{N}_{\text{filter}} - 1)$, where $\mathrm{N}_{\text{filter}}$ is the size of your filter array.

(d) The output $r(t)$ is firing rate. Consider an inhomogeneous poisson process with instantaneous firing rate of $r(t)$. Sample a spike train from this process until 100s.

(e) Compute the spike-triggered average (STA) up to $\mathrm{N}_{\text{filter}}$ time bins before the spikes.

$$\mathrm{STA}(t) = \frac{1}{\mathrm{F}}\sum_f x(t_f - t),$$

where F is the total number of spikes. The STA should look noisy and lack interpretable structure.

(f) Compute the spike-triggered covariance (STC).

$$\mathrm{STC} = \frac{1}{\mathrm{F}}\sum_f \xi_f \xi_f^T$$
$$\xi_f = x(t_f - t) - \mathrm{STA}(t)$$

As a shortcut, you could also utilize the function `numpy.cov(ξ)` to compute the covariance matrix.

(g) Obtain the eigenvectors and eigenvalues of the STC. (You could use the function `numpy.linalg.eig`.) Plot the eigenvalues from high to low. You should see two eigenvalues which are particularly higher than the rest.

(h) Plot the corresponding eigenvectors of these two eigenvalues. They should look similar to the filters $a(t)$ and $b(t)$. The signs can be opposite (the eigenvectors have negative values instead of positive). Why?

(i) How would the model generalize to two pixels (a whitenoise vector $\vec{x}$ with two entries)? How many kernels could there be for n pixels?

2. (GLM-based spike fitting)

The exponential family distribution

$$p(y) = h(y)\, \exp[\theta\, y - A(\theta)]$$

and the (invertible) canonical link function $\langle y \rangle = g^{-1}(\theta)$, with $\theta = \vec{w} \cdot \vec{x}$, define a generalized linear model.

(a) Show that $\vec{w} \cdot \vec{x} = g[\partial_\theta A(\theta)]$.

(b) Find the canonical link function for the Poisson count distribution $p(y) = \frac{(\lambda)^y}{y!}\, e^{-\lambda}$.
Hint: First, identify $h(y)$, θ, and $A(\theta)$.

(c) Let us now consider the following dynamical system (L-NL cascade)

$$\lambda_t = \exp[(s * \phi)_t - (y * \eta)_t]\, \Delta t\ ,$$

in which λ_t is the density of an inhomogeous poisson process that generates spike counts y_t, and $\Delta t = 0.1$ ms. s_t is a stimulus, ϕ_t and η_t are the feedforward and feedback filters, respectively. We now consider the system in discrete time (λ_t is constant in the t-th time bin). Show that the log-likelihood of an output spike (count) train y_t equals

$$\mathcal{L} = \sum_t (y_t \ln \lambda_t - \lambda_t).$$

(d) (Python exercise) Let us consider the two specific kernels

$$\phi_t = a\, t\, \exp[-(tb)^2/2] \qquad \eta_t = d/c \exp[-tc]\mathcal{H}(t)$$

with $\mathcal{H}$ denoting the Heaviside function. Plot the kernels and give biological motivations.

(e) (Python exercise) Load the stimulus s and the measured spike train y from `y.npy` and `s.npy` (Sampling rate 10 kHz) and fit the model parameters by maximizing the likelihood (use `scipy.optimize.fmin`)

(f) Read the paper `Pillow2008.pdf`.

3. (Logistic Regression) Logistic regression is (in its simplest form) a binary classification method ($y = \pm 1$) in which, one models the probability q that an input x is in class $y = +1$ by the logistic function

$$q(x) = [1 + \exp(-wx)]^{-1}\ .$$

The log-likelihood can be written as

$$\ln p(y|x) = \frac{1+y}{2} \ln q(x) + \frac{1-y}{2} \ln[1 - q(x)]$$

(a) Explain the log-likelihood formula.

(b) For multiple observations $(y_i, x_i), i = 1, ..., \ell$ the log-likelihood sums up

$$\ln p(\{y_i\}|\{x_i\})_{i=1,..,\ell} = \sum_i \frac{y_i + 1}{2} \ln q(x_i) + \frac{1 - y_i}{2} \ln[1 - q(x_i)]$$

Show that the gradient can be expressed as

$$\partial_w \ln p(\{y_i\}|\{x_i\})_{i=1,..,\ell} = \sum_i x_i \left[z_i - q(x_i)\right]$$

with $z_i = (1 + y_i)/2$.

(c) Explain the relation of binary logistic regression to GLMs.