Convex Optimization
Project 1

Spring 1402
Due date: 3rd of Ordibehesht

---

1. *Extracting neural signals from an electrophysiological recording.* An electrophysiological (ephys) recording gives the voltage measured by a probe inserted into the brain at regular time intervals, such as every 10^{-4} seconds (0.1 milliseconds). Such recordings capture several neurons firing, as well as noise. In this exercise you will use convex optimization to extract the neural signals, which come from the neurons firing, from the noise in an ephys recording.

   We represent the ephys recording as a vector $y \in \mathbf{R}^T$, with y_t the voltage in time period t, for $t = 1, ..., T$. We model y as

   $$y = s * a + v,$$

   where $*$ denotes convolution, $s \in \mathbf{R}^M$ is the known standard response of a neuron firing (an action potential obtained from the so-called Hodgkin-Huxley model, but you don't need to know that), with M its length, $a \in \mathbf{R}^N_+$ is the unknown activation, and $v \in \mathbf{R}^T$ is the unknown noise signal. The activation signal a is sparse and nonnegative, with $a_t > 0$ meaning that a neuron near the probe fired at time t, with amplitude a_t. We have $T = M + N - 1$, the length of the convolution $s * a$. Written out explicitly the model above is

   $$y_t = \sum_{\tau=1}^{M} a_{t-\tau} s_\tau + v_t, \quad t = 1, ..., T \tag{1}$$

   where we interpret a_t as 0 for $t < 1$ or $t > N$. We refer to $s * a$ as the neural signal.

   We are given y and s, and wish to estimate the activation signal a. (From a we can construct the extracted or cleaned neural signal $s * a$.) To do this we minimize the mean square value of $v = y - s * a$, plus a regularizer, *i.e.*, solve the problem

   $$\begin{array}{ll} \text{minimize} & \frac{1}{T}\|y - s * a\|_2^2 + \lambda r(a) \\ \text{subject to} & a \succeq 0, \end{array}$$

   where $r : \mathbf{R}^N \to \mathbf{R}$ is a convex regularizer function and λ is a positive parameter that scales the regularization. The variable here is a; s and y are given. We let $\hat{a}$ denote a solution of this problem.

   (a) Suggest a regularizer r for which the resulting a would typically be sparse. Then simplify r using the fact that here we have $a \succeq 0$.

   (b) Carry out the method using the regularizer suggested in part (a), on the data given in `neural_signal_data.*`. This defines synthetic data, which includes the true value a^{true}, which of course you would not have in a real problem. You may use the function `visualize_data`, provided with the data, to plot the given y and the true neural signal $s * a^{\text{true}}$. The data gives $T = 2199$ samples, spaced 0.1 milliseconds apart, so the whole recording covers a little more than 0.2 seconds. The true activation a^{true} has 10 nonzero entries.

   Find $\hat{a}$ using $\lambda = 2$. You should find that $\hat{a}$ has around 11 or so nonzero entries (based on a threshold such as a_t ¿ 0.001). Using the function `visualize_estimate` provided with the data, plot the estimated and true activations, $\hat{a}$ and a^{true}, and the estimated and true values of the neural signal, $s * \hat{a}$ and $s * a^{\text{true}}$.

   Use the function `find_nonzero_entries` provided with the data to find the nonzero entries in $\hat{a}$. How well do the nonzero entries in $\hat{a}$ and a^{true} match?

   Hint. CVXPY supports convolution, in its `conv` atom. It returns a $T \times 1$ matrix, so you may need to use `cp.conv(s,a).flatten()` to obtain a vector.

Remark. We do not expect the true and estimated nonzero indices to match exactly; in addition, what is really just one nonzero in the true activation can end up as a few contiguous or nearby nonzero entries in the estimated activation.

(c) *Polishing.* Regularization has the effect of making a smaller than it would be without the regularization. (In statistical estimation this phenomenon is called shrinkage, and is a desirable feature. In this case, it is not.) A method called *polishing* can be used to improve the estimate when this is not desirable.

To do this, we first solve the regularized problem in part (b) above. This gives us $\mathcal{T} = \{\tau | a_\tau > 0.001\}$, the set of times we think a neuron fires (with our threshold). Then we solve the problem again, with *no regularization*, adding the explicit constraint that $a_\tau = 0$ for $\tau \notin \mathcal{T}$.

Carry out this polishing procedure for the $\hat{a}$ found in part (b) to obtain $\hat{a}^{\text{pol}}$. Use the function `visualize_polished` provided with the data to create the same plots as in part (b), *i.e.*, $\hat{a}^{\text{pol}}$ and a^{true} and also $s * \hat{a}^{\text{pol}}$ and $s * a^{\text{true}}$. Does polishing improve your estimate of the neural signal?

Remark. A more sophisticated version of the polishing step can include logic that combines adjacent or nearby nonzero entries in $\hat{a}$ into just one. Specifically, in this problem, biology dictates that neurons cannot be activated twice within 1ms intervals, providing a natural interval to combine adjacent nonzero entries. But we are not asking you to do this.

Good Luck!