A Mixed Integer Quadratically Constrained Representation

Nathan Wycoff

September 10, 2018

1 Review

The Integrate and Fire neuron has the following form, for some neuron, call it n:

$$V'(t) = \sum_{i \in \mathcal{P}_n} w_i \sum_{t_f \in \mathcal{F}_i} \alpha(t - t_f) \tag{1}$$

, with $\mathcal{P}_n$ being the set of presynaptic neurons, that is, neurons connected to neuron n, and $\mathcal{F}_i$ being the set containing the firing times of neuron i. Whenever $V(t) = \nu$, we have that $V(t + \epsilon) \to 0$ as ϵ vanishes. Such t will be called t^a, for "actual firing time".

2 Optimization Approach

The goals are twofold: firstly, to simulate the ODE system, and secondly, given some desired firing time t_i^d, to align actual and desired firing times in some sense, such as by minimizing squared deviations $\sum_i (t_i^d - t_i^a)^2$, leading to a quadratic objective, or absolute deviations, leading to a linear objective.

We would like for the constraints to be simple enough that enormous scaled systems may be solved. In this case, the constraints are simply that the times results from our ODE, that is, that the potential at times t^a (which I am treating as a decision variable) is at the firing threshold ν:

$$V(t_i^a) = \nu \forall i \iff \int_0^{t_i^a} \sum_{i \in \mathcal{P}_n} w_i \sum_{t_f \in \mathcal{F}_i} \alpha(t - t_f) dt = \nu \forall i \tag{2}$$

We will choose the postsynaptic kernel function α, which tells us how an upstream neuron influences a downtstream neuron, to be as simple as possible:

$$\alpha(t - t_f) = \begin{cases} 0 & t - t_f \leq 0 \\ 1 & 0 \leq t - t_f \leq \tau \\ 0 & t - t_f \geq \tau \end{cases}$$

that is, it's simply the characteristic function of the interval $[0, \tau]$, where τ is a constant (for now assumed known) giving the firing duration.

The integral is therefore simply a piecewise linear function, and we obtain the following constraint (for simplicity, we will now concern ourselves only with two neurons, and the potential of the upstream neuron. Each neuron will only fire once. These assumptions are not without loss of generality, as we will see later):

$$w(t^a - t_f)\mathbf{I}_{[t^a - t_f \in [0,\tau]]} + \tau \mathbf{I}_{[t^a - t_f > \tau]} = \nu \tag{3}$$

With the introduction of three binary variables, this constraint can be quadraticized.

$$-Lb_1 \leq t^a - t_f \tag{4}$$

$$Lb_2 \geq t^a - t_f \tag{5}$$

$$Lb_3 \geq t^a - t_f - \tau \tag{6}$$

$$b_1 + b_2 + b_3 = 1 \tag{7}$$

$$b_2(t^a - t_f) + b_3\tau = \mu\nu \tag{8}$$

where $\mu = 1/w$.

Solving the forward problem is equivalent to finding a feasible solution, while solving the inverse problem is finding the optimal solution. I successfully implemented this 1 neuron program in Julia using JuMP and Gurobi.

3 Challenges of Scale

This model has a lot of potential because it allows us to symoultaneously solve the forward and inverse problems. However, it's not clear to me how to scale this beyond 1 neuron.

The difficulty comes in with the w term in the constraint, which is forming a 3rd degree term before I reparameterize it with μ. This is no longer possible with more than 1 neuron. However, we should be able to solve the forward model, as w is fixed in that case. This would give us sensitivities we need to do gradient descent.

Thoughts:

1. If we fix all but 1 weight for each incoming connection, we may be able to solve the program, and find an overall solution by alternating which weight is free. Such an approach is best for models with few incoming connections, which is not the case for popular neural archetectures. But maybe we can develop new architectures?

2. It may be possible to have a different τ for each connection and optimize those instead of the weights.

3. Constraining the weights to be binary may be interesting