Attempt to solve non-linear differential equation models to predict system responses

Tomas Ukkonen, Novel Insight, 2022

tomas.ukkonen@iki.fi

Predicting brain EEG responses might benefit from differential equation model which predicts what happens next in the brain normally. The differential equation model uses neural nets as non-linear function.

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{z}), \text{ where } \boldsymbol{z} \text{ is outside stimulus, } \frac{d\boldsymbol{z}}{dt} \approx \text{small}$$

This may be a bit difficult problem so we initially simplify the problem.

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x}(t)), \text{ where } \boldsymbol{f}(\boldsymbol{x}) \text{ is neural net}$$

This differential equation model can be easily numerically simulated using, for example, Runge-Kutta integration method $\boldsymbol{x}(t) = \boldsymbol{x}_0 + \int \boldsymbol{f}(\boldsymbol{x}(t)) dt$.

MSE minimization

For fitting neural network parameters $\boldsymbol{w}$ to data $\{(\boldsymbol{y}_i, t_i)\}$, we can minimize for least squares:

$$e(\boldsymbol{w}) = \frac{1}{2N} \sum_{i=1}^{N} (\boldsymbol{x}(t_i, \boldsymbol{w}) - \boldsymbol{y}_i)^T (\boldsymbol{x}(t_i, \boldsymbol{w}) - \boldsymbol{y}_i)$$

$$\nabla_{\boldsymbol{w}} e(\boldsymbol{w}) = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}(t_i, \boldsymbol{w}) - \boldsymbol{y}_i)^T \int_0^{t_i} \nabla_{\boldsymbol{w}} \boldsymbol{f}(\boldsymbol{x}(t)) dt$$

This is difficult to solve(?) but we can use Runge-Kutta to solve for $\boldsymbol{x}(t_i)$ terms. After this we can do Runge-Kutta for each weight $\boldsymbol{w}_k$ in gradient term $\nabla_{\boldsymbol{w}} \boldsymbol{f}(\boldsymbol{x}(0))$ and substract $\boldsymbol{x}_0$ term from the results to calculate the gradient matrix term(?).

Bayesian Hamiltonian Monte Carlo sampler

Because it is difficult to use Runge-Kutta repeatedly. It might be easier to just use Runge-Kutta once to estimate values $\boldsymbol{x}(t_i)$. We can then define gaussian error function with zero mean and unit variance error term. This means we have a model $p(\text{data}||\boldsymbol{w}) \approx \text{Exp}(-\text{error}(\boldsymbol{w}))$ which we can reverse by using bayesian rule and assume flat constant prior for weight values. This is then maximum likelihood of data method.

Once we have defined probability distribution we can select random starting point $\boldsymbol{w}_0$ near origo and sample using Hamiltonian Monte Carlo sampler and select maximum probability weight $\boldsymbol{w}$ that is found.

The exact probability function is:

$$p(\boldsymbol{w}||\text{data}) \propto p(\text{data}||\boldsymbol{w}) \propto \text{Exp}\left(-\frac{1}{2N} \sum_{i=1}^{N} (\boldsymbol{x}(t_i, \boldsymbol{w}) - \boldsymbol{y}_i)^T (\boldsymbol{x}(t_i, \boldsymbol{w}) - \boldsymbol{y}_i)\right)$$

And we use Runge-Kutta to simulate $\boldsymbol{x}(t_i)$ values every time $\boldsymbol{w}$ changes. In practice, we need also match time values t_i to Runge-Kutta step intervals and we either linearly interpolate $\boldsymbol{x}$ values from too closest time steps or select $\boldsymbol{x}$ which time value is the closest (initially try this).

NEWS: HMC sampler cannot find good results but 11-layer deep neural net's error becomes slowly smaller (several hours) so it could maybe find something.

Expectation Maximization

Expectation maximization techniques are recommended for the problem. Find more information.

PLAN

More study.