Exploring the Modelling of User Arrival Times of Online Streaming Sites

Nicolas S. Blumer [1,2,3] Supervised by Prof. Ritabrata Dutta [1] and Shreya Sinha Roy [1]

[1]Department of Statistics, University of Warwick [2]Department of Computer Science, ETH [3]Department of Law, University of Zurich

Introduction

We aim to generate synthetic user data that mimics real-world interactions to experiment with recommender systems.
Given the last interaction time t_i and interactions r_i generate predictions for the next time the user visits the site/app. $t_{i+1} \sim f_\theta(\cdot|t_i, r_i)$. Our idea is that users have a hidden state that evolves over time and dictates when users interact with the system.

ContentWise Dataset

User data from a streaming site. Comprises of user interactions (positive and negative impressions) and additional metadata (time-steps, item IDs). After processing the data we obtain:

Timestamps	User	Items	Reward	Means	Log Variances
[1.54, 13.3]	34567	[[1, 17],[2,3,16]]	[[1,0],[0,0,0]]	[0.1, -0.15]	[0.04, 0.13]
[13.4, 31.0, 54.2]	34568	[[1],[0,12], [19]]	[[0],[0,0], [1]]	[-0.2, -0.69]	[4.20, 0.42]

Model as Partially Deterministic Markov Process (PDMP)

- PDMP Model 1: Flow Function(Φ): Evolution of the users hidden state (X) while not interacting with the system.

$$X_{t+h} = \phi(X_t, h), \qquad h \in [0, +\infty).$$

We consider ϕ to be an ODE (model via a neural ODE f): $X_{t+h} = X_t + \int_0^h f(X_v)dv$.

- PDMP Model 2: Intensity function (λ): For each state X, there is a random waiting time T_X at the end of which the process jumps. The distribution of T_X depends on λ through the following formula

$$P(T_X > h|X_t) = \exp\left(-\int_0^h \lambda(\phi(X_t, v))dv\right).$$

- PDMP Model 3: Given the hidden state of user u at time t: X_t, predict the reaction to recommendations.
- PDMP Model 4:Transition Kernel (K): Captures the impact of the finished interaction on the state of the user. Creates a jump in the hidden state of the user.

$$X_t^+ = K(r_t, X_t)+z, \qquad z \sim \mathcal{N}(0, \Sigma).$$

Generating samples: Invert the cdf (Model 2) to find h. Requires solving for h.

Prequential Energy Score: An estimate of the energy score:

$$\hat{S}(\mathbf{x}, y) = \frac{2}{M}\sum_{i=1}^{M}\|x^{(i)} - y\|_2^\beta - \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M}\|x^{(i)} - x^{(j)}\|_2^\beta.$$

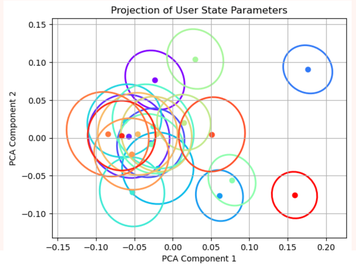
where: $\mathbf{x} = (x^{(1)}, x^{(2)}, \dots x^{(M)})$ is a vector of samples from the predicted distribution, y is the observed or true value, M is the number of samples, $\beta \in (0, 2)$. To get a prequential scoring rule, the model is conditioned on the previous ground truth sample(s): $P_{t+1}^\theta(\cdot|y_t)$.

Findings

- Pytorch **ODE solvers fail** due to numerical issues, but due to the properties of the functions, you can turn solving for h into a simpler problem that requires one for-loop.
- The model **can't stop generating** new arrival times.
- The model is very **computationally expensive**.
- The problem statement is **not differentiable** with respect to the parameters because we need to solve for h. Evolutionary Strategies (ES) as an alternative is too inefficient.

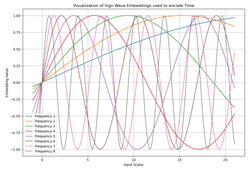
Generating Users: Variational Auto-Decoder (VAD)

Each user (20 in our experiments) is modeled with a normal distribution of possible states at t=0. The distribution's parameters are optimized during training, with the KL divergence from a standard Gaussian (mean 0, variance 0.01) added to the loss.
New users can be generated by sampling from this distribution, akin to Variational Auto-Encoders (VAE).



Wave Encoding of Time

In order to help the model better learn how to interpret time, times were encoded into several sine and cosine waves with different frequencies.
This led to the models being able to learn fast changes in the outputs.



Alternative Model 1: Function Approximation

- FA Model 1: MLP: (state (X_{t_i}), noise (z_i)) $\rightarrow$ waiting time (h_{i+1}).
  M simulations of the waiting time: $\hat{h}_{i+1}^j = f_{\theta_1}(X_{t_i}, z_i^j) \quad j = 1, 2, \dots M.$
- FA Model 2: MLP: (state (X_{t_i}) , waiting time (h_{i+1}) $= t_{i+1} - t_i$)$\rightarrow$ new State ($X_{t_{i+1}}$).

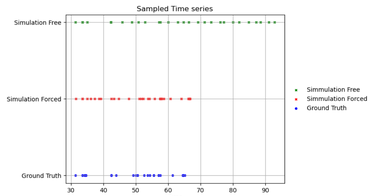
$$X'_{t_{i+1}} = f_{\theta_2}(h_{i+1}, X_{t_i}).$$

- FA Model 3(Jump Model): MLP: reaction to recommendations (r_i) $\rightarrow$ jump delta (Δ_i)

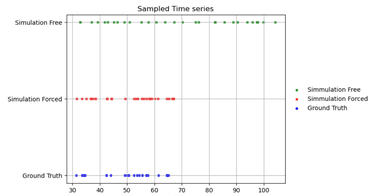
$$X_{t_{i+1}} = \Delta_i + X'_{t_{i+1}}, \qquad f_{\theta_3}(r_i) = \Delta_i.$$

Optimize the models by generating several waiting times $\hat{h}_{i+1}^j$ and compare them to the true h_i vie the prequential energy score.

Experiments



Model without jump



Model with jump

Findings

- The model is faster to train, but still needs batch size = 1.
- The model still can't stop generating data.
- The predicted time intervals between the events tend to be too high.

Alternative Model 2: Density Estimation

- DE Model 1: MLP: (state (X_t), waiting time (h)) $\rightarrow$ new state(X_{t+h}).

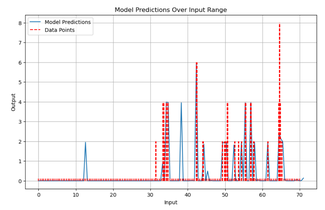
$$f_{\theta_1}(X_t, h) = X_{t+h} = X_{t'}.$$

- DE Model 2: MLP or NODE: state($X_{t'}$) $\rightarrow$ intensity of events($\hat{\lambda_{t'}}$).

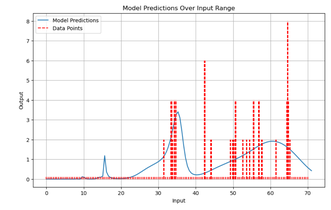
$$f_{\theta_2}(X_{t'}) = \hat{\lambda}_{t'}.$$

The model is trained by providing the times when an event happens and uniformly sampled times where no interaction occurs. The ground truth λ's are obtained by performing a density estimation of the event times using a (Rectangular) Kernel and optimized via the MSE.

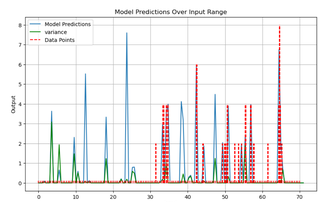
Experiments



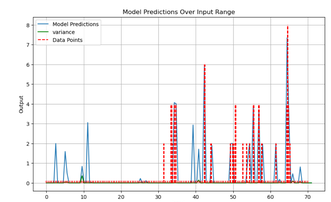
MLE



MLE without Time Embedding



Variational Inference for Model 1 and 2



Variational Inference for Model 2

Findings

- This method is easier to parallelize and generally more efficient.
- The model knows when not to generate samples.
- Unsure how to incorporate user interactions.

Conclusions

Learnings

- Don't neglect computational costs.
- Neural ODEs are usually very slow, and hard to train.
- Training complex models on small datasets often leads to difficulties when optimizing.
- Density estimation could be a reasonable alternative problem formulation.

Future Work

- Extend experiments to more users.
- Model the user's reactions to items.
- When using the BBC data, incorporate additional information such as genres into the model.
- Incorporate interactions (jumps) into the density estimation model and use uncertainty estimates.

nicolas.blumer@warwick.ac.uk