Report for Point Process Project

Zhaoyu Li

June 30, 2020

1 Introduction

With the development of deep learning, the neural temporal point process is becoming more and more popular. Neural point process model is good at prediction, while it is weak on interpretability. The project is to build a Interpretable Neural Temporal Point Process model, which is good at prediction and interpretability simultaneously. In this project, based on the paper "Improving Interpretability and Predictive Performance of Neural Temporal Point Processes", I implement INTPP model to strengthen the understanding of neural point process and bridge between statistical point process and neural point process. I also conduct experiments on both synthetic sequences which are generated in homework1 and ATMs dataset.

2 Prerequisites

I implement the project using Python API. To run my code, **pytorch (version==1.1.0)**, **sklearn**, **easydict** and **pickle** should be installed. All the parameters are stored in **utils/config.py**. You can use your own parameters by changing **experiment*.yaml** to cover the default parameters if you like. Run **python train_eval_synthetic.py – cfg (experiment0.yaml or experiment1.yaml)** to use INTPP model to fit on synthetic dataset which generates 10 or 2 dimensional data. Run **python train_eval_atm.py – cfg experiment2.yaml** to use INTPP model to train on ATMs training dataset and test on test dataset as well. The results are showed in Section 4.

3 Encountered Problems

The implementation of INTPP is quite hard project for me, especially to reproduce the same results showed in the paper. It takes me almost a week to finish it. When trying my first version of implementation, I encounter some problems as below:

- What's the nerual network architecture of INTPP and what's the difference between it and RMTPP?
- How to write the loss of INTPP?
- How to predict the next event time for multi-dimensional data?
- How to calculate A?

For first problem, it seems like the INTPP model's architecture is same as RMTPP. However, there is a little different. For INTPP, there is no linear function and activate function to embed the hidden state from RNN output. Thanks to the code from Liangliang Shi, I translate tensorflow implementation of INTPP loss to pytorch one and use the same parameters in the code of newton iteration. After asking to TA Yijun Wang and Liangliang Shi as well as discussing with Ziyu Wang, I understand how to predict the next event time by summing the row of l_j and c to present the intensity function. Also given the code of Liangliang Shi, I understand how to calculate A.

However, when I run my code without error, there are some other problems:

- It's not stable for the parameters to converge in synthetic dataset.
- The time prediction for ATMs dataset is bad.

For the first problem, I use a small grid search to select parameters, use LR scheduler to reduce LR and stop using event loss to make the training stage stable. Also note that in order to train the dataset, I clip the event sequences and time sequences into pieces. However, there are partial bad sequences in the training stage due to the clipping, so I discard the sequences whose interval is too large. It improves the time prediction a lot. I also find it's because some summation of rows of l_j is negative, which makes the time step prediction not stable.

4 Results and Analysis

In this project, I use different metrics for synthetic dataset and ATMs dataset. For synthetic dataset, I use mean absolute error (MAE) of hawkes parameters. For ATMs dataset, I use two metrics to measure the performance of my model. The first one is "Event Type Prediction", which calculates precision, recall, macro-F1 score over 7 event types and then average over all types. The second one is "Event Time Prediction", which is the MAE that measures the absolute difference between the predicted time point and the actual one.

For 10-dimensional synthetic dataset in homework 1, the MAE of μ and A are showed as below:

Table 1: MAE of μ

0.001	2	0.38	0.111	0.36	0.4	0.81	0.667	0.75	0.75

Table 2: MAE of A

0.278	0.658	25.38	0.058	24.03	0.086	0.531	0.031	0.518	0.460
0.079	0.381	0.026	0.047	3.003	0.561	0.040	0.097	0.868	0.012
0.824	0.042	3.18	0.5883	0.011	0.012	0.630	0.296	0.097	0.101
0.246	0.390	0.055	0.194	0.526	0.018	0.034	0.100	0.022	0.023
0.870	0.725	0.001	0.005	0.135	0.001	0.042	0.773	3.839	0.002
0.579	0.081	0.031	0.026	0.049	0.487	0.918	0.530	0.510	0.600
0.115	0.943	8.427	0.060	0.467	0.070	2.864	0.652	0.985	0.450
0.033	0.343	0.049	0.582	0.050	0.376	0.113	0.342	0.558	0.035
0.125	0.143	0.565	0.052	0.715	0.130	0.081	0.120	0.721	0.797
0.608	0.024	0.217	0.035	2.736	0.117	1.280	0.035	0.066	0.821

For 2-dimensional synthetic dataset, the MAE of μ and A are showed as below:

Table 3: MAE of μ

0.031	0.005

Table 4: MAE of A

0.006	0.159
0.001	0.122

For 10-dimensional dataset, since the weight decay $w = 1$ is quite small, the previous events' influence is large, which makes the training not very stable. Since I calculate A for each batch data and average all the batch data to get the final A and the difference and variance between each batch data is quite large, the final A may be not very accuracy. Also note that there are some very small item in matrix A, e.g. 0.001, so a little error may result in a big MAE and that's why there are some elements' MAE larger than 1. But for 2-dimensional dataset with $w = 1$, all elements of μ and A are in $[0.2, 0.7]$, the training is stable and the preformance is quite well.

For ATMs dataset, the MAE of event time prediction is **0.218** and the results of event type prediction are showed as the Table 5 below:

Table 5: Event Type Prediction of INTPP model

	PRT	CNG	IDC	COMM	LMTP	MISC	TIKT	avg
Precision	0.946	0.975	0.857	0.905	0.938	0.986	0	0.801
Recall	0.858	0.986	0.876	0.962	0.944	0.980	0	0.801
macro-F1 Score	0.900	0.980	0.866	0.933	0.941	0.983	0	0.801

As for the result, the prediction of error type TIKT is very bad since that there is few training samples in the training dataset and the appearance of TIKT is irregular that it's hard for the model to predict. On the opposite, the prediction of CNG, COMM, LMTP and MISC is quite well. It is because they often continuously appear in an event sequence and the model may learn the pattern of their appearances very well. To address this issue, I think maybe use some weight sampling methods may improve the performance of TIKT.