Purchase Prediction based on Recurrent Neural Networks with an Emphasis on Recent User Activities

Quanyu PIAO (Waseda Univ.), Joo-Young LEE (Wider Planet, Inc.), Tetsuya SAKAI (Waseda Univ.)

DEIM 2020 B4-3

Contents

1. Introduction

- An E-Commerce prediction task
  - E-Commerce is common nowadays. How to use these business data for better profit is important.
  - Operators collect users click data.
    - Tremendous users browse EC site anytime.
      - The data is **very big**.
  - Characteristics
    - **Sequentiality**
    - **Uncertainty**
    - **Multidimensionality**

1. Introduction

- User data: Session
  - User click data
    - Session ID, Timestamp, Item ID, Category, etc.

SSID	Timestamp	Item ID	Category
1	2014-04-07T10:51:09.277Z	214536502	0
1	2014-04-07T10:54:09.868Z	214536500	0

  - User buy data
    - Session ID, Timestamp (omitted), Item ID, Price, Quantity, etc.

SSID	Item ID	Price	Quantity
420374	214537888	0	1
420374	214537850	0	1

1. Introduction

- Our task
  - Aggregate user data with some specific methods.
  - Show and compare the performances of the models trained by original data and aggregated data.
- Our goal
  - Main
    - Avoid the models overly concentrating on specific user activities.
  - Sub
    - Reduce user data size.
    - Accelerate the model training process.

1. Introduction

- Previous works
  - Random Forest
  - Gradient Boosting Decision Tree (GBDT)
  - Factorization Machine (FM)
- Our model selection
  - Neural Network
    - Recurrent Neural Networks (RNN): Adapt sequential data well.
      - **Long Short-Term Memory (LSTM)**

2. Methodology

- Data Aggregation
  - Start point
    - Analogous to the way people recall the products they browsed in online shopping.
  - We believe that the aggregation can
    - Summarize and also compress user data.
  - It is hard to give prediction for short sessions.
    - The overall record length will be significantly reduced after aggregation, which may give more attention to short sessions for better prediction.

2. Methodology

- Aggregation definitions
  - We define number of activities we aggregated within one session in every step P_i as:

$$P_i, \qquad i = 1,2,3, \ldots$$

  - Then, the total numbers of activities aggregated $N_{aggregated}$ can be represented as:

$$N_{aggregated} = \sum_{i=1}^{\max(i)} P_i\,, \qquad i = 1,2,3, \ldots$$

  - Obviously, $N_{aggregated}$ should be smaller than the longest session in the dataset:

$$N_{aggregated} \leq maxLength(session)$$

2. Methodology

- **Natural Aggregation**
  - We aggregate numbers of activities in natural number sequence:

$$P_i = i, \qquad i = 1,2,3,\ldots$$

- **Fibonacci Aggregation**
  - The Fibonacci number sequence can be represented as:

$$F(i) = F(i-1) + F(i-2), \qquad i = 1,2,3,\ldots$$

  - So, the number of aggregated activities in every step will be:

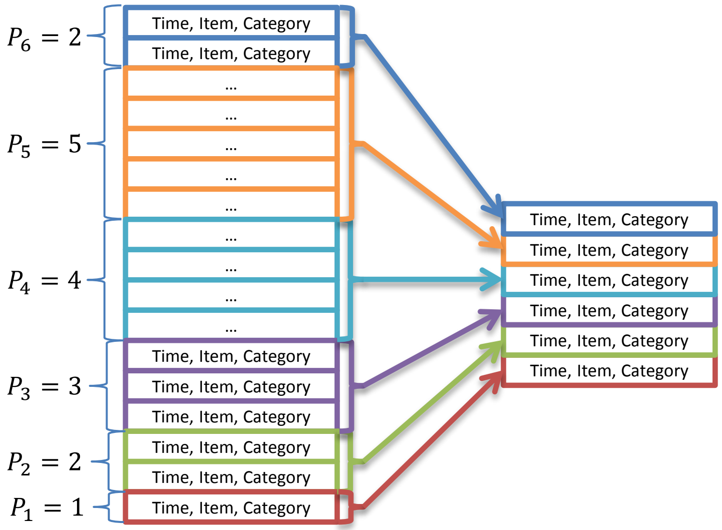
$$P_i = \begin{cases} 1, & i = 1 \; and \; 2 \\ F(i), & i = 3,4,5,\ldots \end{cases}$$

- **Exponent Aggregation**
  - Similarly way. We choose the base as 2.

$$P_i = 2^{i-1}, \qquad i = 1,2,3,\ldots$$

2. Methodology

- Natural Aggregation



2. Methodology

- Aggregation method for features in every step
  - For numerical features, e.g., timestamp,

$$feature_{new} = avg(\sum feature_{original})$$

  - For non-numerical features, e.g., item ID and category, we choose the most clicked item and its category in this aggregating step.
    - We should keep the correlativeness of the features.

$$feature_{newItem} = mostClicked(feature_{clickedItemList})$$
$$feature_{newCategory} = Category(feature_{newItem})$$

2. Methodology

- Embedding
  - We embed all features into low-dimensional dense vector spaces.
- Long Short-Term Memory (LSTM)
  - A popular model for sequential data processing.
  - Especially, we use bi-directional LSTM, which is Bi-LSTM.
    - We believe that the Bi-LSTM model can get more user action information with chronological view and inverse-chronological view.

3. Experiments

- Dataset
  - The RecSys 2015 Challenge dataset published by YOOCHOOSE.

Total clicks	Click sessions	Buy sessions	Unique Item	Unique Category
33,003,944	9,249,729	509,696	52,739	339

- Data preparation
  - Data imbalance
    - Create balanced data
  - Select sessions with suitable length
    - With padding

Aggregation	Activity Length Limit	Shrunken Percentage
Natural	36	99.60%
Fibonacci	33	99.46%
Exponent	31	99.35%

3. Experiments

- Data properties processing
  - Click data: Unchanged.
  - Buy data: Only session ID, timestamp and item ID are retained.
  - Timestamp: Split into **month**, **day**, **day of week**, **hour,** minute and second.
  - Category ID: Unchanged
  - Target data (label): For every session in click data, if the session ID exists in the buy data, this means the session is a buy session (1). Otherwise, it is a not-buy session (0).

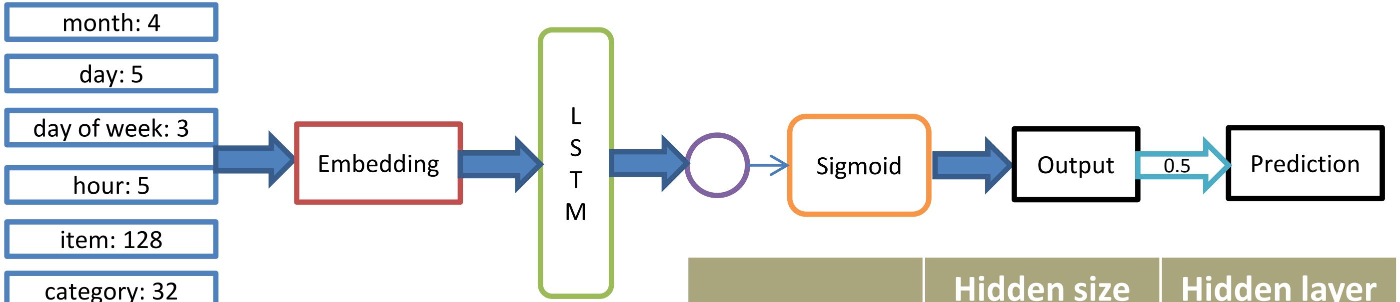
3. Experiments

- Feature data after preprocessing
  - [SSID]
  - [month:day:day of week:hour:minute:second:item ID:category ID|…]
  - [0 or 1]

SSID	Feature	is_buy
7	4:2:3:6:38:53:389:0|4:2:3:6:39:5:847:0	0
367	4:7:1:9:42:12:566:0|4:7:1:9:43:14:566:0	1

3. Experiments

- Activity Aggregation
  - We apply the three aggregating methods to create aggregated datasets.
- Model



	Hidden size	Hidden layer
Simple	32	1
Complex	128	3

3. Experiments

- Results of balanced test dataset (buy: not-buy = 1:1)

	Buy Precision	Not-buy Precision	Buy Recall	Not-buy Recall	ROC_AUC
Simple	**0.7330**	0.7387	0.7363	**0.7354**	**0.7358**
Simple with Natural Aggregation	0.6645	0.7731	0.8247	0.5892	0.7070
Simple with Fibonacci Aggregation	0.7086	0.7384	0.7528	0.6927	**0.7228**
Simple with Exponent Aggregation	0.6419	**0.7816**	**0.8492**	0.5326	0.6909
Complex	0.7178	**0.7476**	**0.7585**	0.7058	**0.7322**
Complex with Natural Aggregation	**0.7414**	0.6676	0.5994	**0.7937**	0.6965
Complex with Fibonacci Aggregation	0.7029	0.7270	0.7390	0.6899	**0.7145**
Complex with Exponent Aggregation	0.7324	0.6817	0.6352	0.7709	0.7031

3. Experiments

- Results of original test dataset (buy: not-buy = 1:18)

	Buy Precision	Not-buy Precision	Buy Recall	Not-buy Recall	ROC_AUC
Simple	**0.1375**	0.9795	0.7339	**0.7343**	**0.7341**
Simple with Natural Aggregation	0.1039	0.9829	0.8221	0.5906	0.7064
Simple with Fibonacci Aggregation	0.1249	0.9797	0.7503	0.6959	**0.7231**
Simple with Exponent Aggregation	0.0957	**0.9842**	**0.8528**	0.5328	0.6928
Complex	0.1295	**0.9806**	**0.7582**	0.7058	**0.7320**
Complex with Natural Aggregation	**0.1429**	0.9714	0.5948	**0.7941**	0.6945
Complex with Fibonacci Aggregation	0.1216	0.9786	0.7385	0.6915	**0.7150**
Complex with Exponent Aggregation	0.1390	0.9734	0.6368	0.7713	0.7040

3. Experiments

- Data compress rate and training time reduction rate (Approximation)

	Dataset compressed rate	Training time reduction rate
Simple	20%	15%
Complex		50%

4. Conclusions

- We define three kinds of data aggregation methods.
  - Natural, Fibonacci, and Exponent Aggregation
- *Except **Fibonacci Aggregation***
  - For simple LSTM model trained by aggregated data:
    - Better: Not-buy Precision, Buy Recall
    - Worse: Buy Precision, Not-buy Recall
  - For complex LSTM model trained by aggregated data :
    - Better: Buy Precision, Not-buy Recall
    - Worse: Not-buy Precision, Buy Recall

4. Conclusions

- **Fibonacci Aggregation**
  - Performs similar with the results trained by original data.
  - Gives the best comprehensive performance of the three aggregating methods.
    - With 97.5% ~ 98.5% of the best performance, it provides data compressing and huge training acceleration.
- The test results for balanced and original dataset are similar.

5. Future work

- Overfitting problem
  - We may need better model and adjustment of hyperparameters.
  - Apply dynamic length input to the model (dynamic RNN).
- For better performance
  - Do more feature analysis.
    - Feature selection, combination, etc.
- Explore better solutions for more normalized and imbalanced real-life data.

Reference

[1] Breiman, Leo. “Random Forests.” Machine Learning 45 (2001): 5-32.

[2] Friedman, Jerome H.. “Greedy function approximation: A gradient boosting machine.” (2001).

[3] Rendle, Steffen. “Factorization Machines.” 2010 IEEE International Conference on Data Mining (2010): 995-1000.

[4] Rendle, Steffen. “Factorization Machines with libFM.” ACM TIST 3 (2012): 57:1-57:22.

[5] Wu, Zhenzhou et al. “Neural Modeling of Buying Behaviour for E-Commerce from Clicking Patterns.” RecSys ’15 Challenge (2015).

[6] Ben-Shimon, David et al. “RecSys Challenge 2015 and the YOOCHOOSE Dataset.” RecSys ’15 (2015).

[7] Sheil, Humphrey et al. “Predicting Purchasing Intent: Automatic Feature Learning using Recurrent Neural Networks.” ArXiv abs/1807.08207 (2018): n. pag.

[8] Gligorijevic, Djordje et al. “Time-Aware Prospective Modeling of Users for Online Display Advertising.” ArXiv abs/1911.05100 (2019): n. pag.

[9] Mikolov, Tomas et al. “Efficient Estimation of Word Representations in Vector Space.” CoRR abs/1301.3781 (2013): n. pag.

[10] Liu Zhiyuan,Sun Maosong,Lin Yankai, et al. “Knowledge Representation Learning: A Review[J].” Journal of Computer Research and Development, 2016, 53(2): 247-261.