Recurrent Neural Network based Recommender System

Yongchan Choi

26, Dec, 2017

Outline

- Introduction : Recurrent Neural Network
- Session-based Recommendation with RNN
- Improved RNN for session-based Recommendation
- RRN
- Joint training of ratings and reviews with RRN

RNN

- devised to model variable-length sequence data
- existence of an internal hidden state

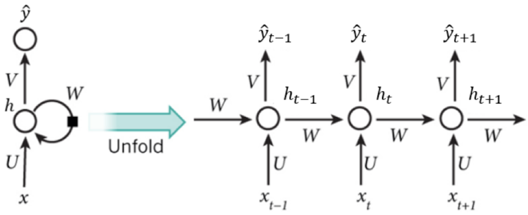

Figure: RNN structure

RNN

- Share parameters W, V, U
- Update parameters using BPTT(Backpropagation through time)
- Vanishing gradient problem (i.e difficult to keep long term memory)
  - LSTM, GRU

Session-based Recommender with RNN

- Many e-commerce recommender systems and most of news and media sites do not typically track the user-id
- Neighborhood methods are based on co-occurrences of items in sessions.
- But, Neighborhood methods cannot consider a sequential data.

Session-based Recommender with RNN

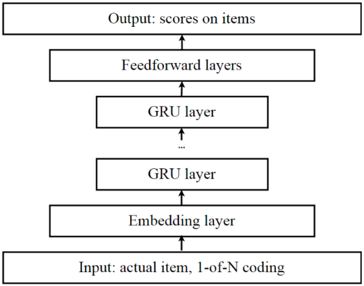

Figure: The Structure of Session-based Recommender with RNN

- input : actual state of the session (1-of-N encoding)
- output : item of the next event in the session

Session-based Recommender with RNN

Practical points

- Session-Parallel mini-batch
- Sampling on the output
- Ranking loss

Session-based Recommender with RNN

Session-Parallel mini-batch

1. Create an order for the sessions
2. Use the first event of the first X sessions to form the input of the first mini-batch
3. 2nd mini-batch is formed from the second events and so on
4. If any sessions end, the next available session is put in its place

- Sessions are assumed to be independent
- reset the appropriate hidden state when switch occurs

Session-based Recommender with RNN

Session-Parallel mini-batch

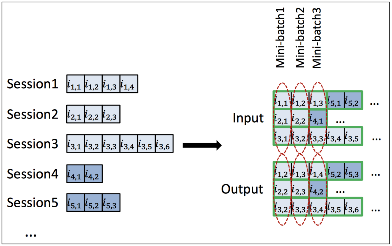

Figure: Session-Parallel mini-batches

Session-based Recommender with RNN

Sampling on the output

- Calculating a score for each item in each step would make the algorithm scale with the product of the number of events.
- Unusable in practice
- Sample the output and compute the score for a small subset of the items
- Only some of the weights will be updated
- need to compute scores for some negative examples and modify the weighs so that the desired output is highly ranked

Session-based Recommender with RNN

Ranking loss

- Pointwise ranking was unstable in this network
- Use pairwise ranking loss
- $L_s = -\frac{1}{N_s}\sum_{j=1}^{N_S} log(\sigma(\hat{r}_{s,i} - \hat{r}_{s,j}))$
  where N_S is the sample size, $\hat{r}_{s,k}$ is the score on item k at the given point of the session. i is the desired item(next item in the session) and j are the negative samples
- General ranking loss : $\frac{1}{N_s}\sum_{j=1}^{N_S} I(\hat{r}_{s,j} > \hat{r}_{s,i})$
- To avoid unstability, Adding regularization term to the loss
- $L_s = \frac{1}{N_s}\sum_{j=1}^{N_S} \sigma(\hat{r}_{s,j} - \hat{r}_{s,i}) + \sigma(\hat{r}_{s,j}^2)$

Improved RNN for session-based Recommendation

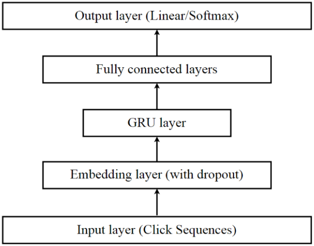

Figure: The structure of Improved RNN for session-based Recommendation

- input : $\mathbf{x} = [x_1, x_2, ..., x_n]$ where $x_i \in \mathbb{R} \quad (1 \leq i \leq n)$
- output : $\mathbf{y} = [y_1, y_2, ..., y_m] \in \mathbb{R}^m$ where m is the number of items

Improved RNN for session-based Recommendation

- Let x_{r+1} be the next click of the click sequence $\mathbf{x}$
- Represent $V(x) \in \mathbb{R}^m$ as 1-Hot encoded vector
- Use loss $L(M(\mathbf{x}), V(x_{r+1}))$
  where L is cross-entropy and $\mathbf{y} = M(\mathbf{x})$

Improved RNN for session-based Recommendation

Data augment

- Given an input training session $[x_1, x_2, ..., x_n]$, Generate the sequences and corresponding labels $([x_1], V(x_2)), ([x_1, x_2, V(x_3)), ..., ([x_1, x_2, ..., x_{n-1}], V(x_n))$ for training
- Embedding dropout
- Intuitively, Users may have accidentally clicked on items that are not of interest
- Delete clicks randomly

Improved RNN for session-based Recommendation

Data augment

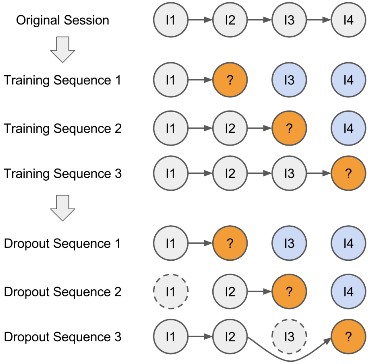

Figure: Embedding droupout

Improved RNN for session-based Recommendation

Adapting to temporal chages

- Learning a recommendation model on the entire dataset may lead to worse performance since the model ends up focusing on some out-of-date properties
- Use entire train data for pre-training

Improved RNN for session-based Recommendation

Use of privileged information

- The item sequence clicked by user after an item may also contain information about that item
- Denote $\mathbf{x}^* = [x_n, x_{n-1}, ..., x_{r+2}]$ where n is the length of the original session
- Minimize a loss of the form :
  $(1-\lambda)L(M(\mathbf{x}), V(x_n)) + \lambda L(M(\mathbf{x}), M^*(\mathbf{x}^*))$
  where $\lambda \in [0, 1]$ is a tradeoff parameter

Recurrent Recommender Networks

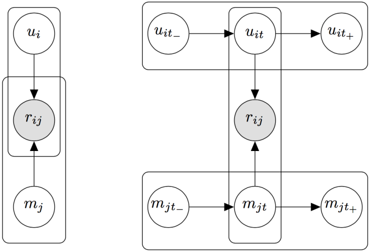

Figure: Left : time-independent Recommendation, Right : time-dependent Recommendation

Recurrent Recommender Networks

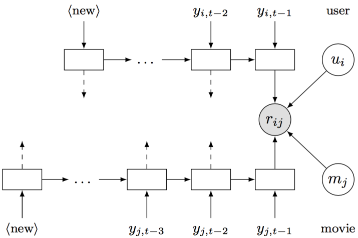

Figure: The structure of Recurrent Neural Networks

Recurrent Recommender Networks

- input : $y_t = W_{embed}[x_t, 1_{newbie}, \tau_t, \tau_{t-1}]$
  where $x_t \in \mathbb{R}^M$, M : the number of movies,
- u_i, m_j : latent vectors for user, movie
- output $\hat{r}_{i,j|t}$: the estimated rating of user i, movie j
- $u_t = LSTM(u_{t-1}, y_t)$
- output : $\hat{r}_{i,j|t} = f(u_{it}, m_{jt}, u_i, m_j) =< \widetilde{u}_{it}, \widetilde{m}_{jt} > + < u_i, m_j >$
  where $\widetilde{u}_{it}, \widetilde{m}_{jt}$ are affine function of u_{it}, m_{jt}

Recurrent Recommender Networks

- $minimize_{\theta} \sum_{(I,j,t) \in I_{train}} .(r_{i,j|t} - \hat{r}_{i,j|t}(\theta))^2 + R(\theta)$
- Parameter update using Subspace descent strategy

Joint training of ratings and reviews with RRN

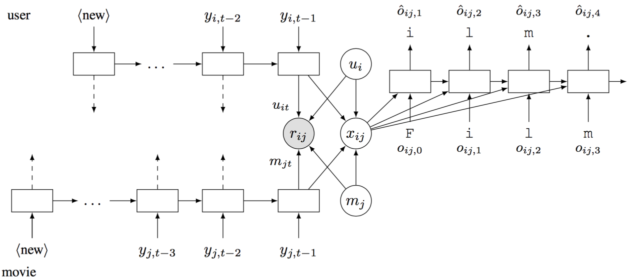

Figure: The structure of JTRRRRN

Joint training of ratings and reviews with RRN

- $x_{joint,ij} = \phi(W_{joint}[u_{it}, m_{jt}, u_i, m_j] + b_{joint})$
- and $\widetilde{x}_{ij,k} = [x_{o_{ij,k}}, x_{joint,ij}]$
  where ϕ is some non-linear function.
- $h_{ij,k} = LSTM(h_{ij,k-1}, \widetilde{x}_{ij,k}$ and
  $\hat{o}_{ij,k} = softmax(W_{out} h_{ij,k} + b_{out})$

Joint training of ratings and reviews with RRN

- $L = \sum_{i,j \in I_{train}} [(\hat{r}_{ij}(\theta) - r_{ij})^2 - \lambda \sum_{k=1}^{n_{ij}} log(Pr(o_i = ij, k|\theta))]$
- where I_{train} is the training set of (i, j) paires, n_{ij} is the number of characters in the review user i gives to movie j
- The review can be viewed as auxiliary task to facilitate rating prediction