ADER: Adaptively Distilled Exemplar Replay Towards Continual Learning for Session-based Recommendation

...and other methods

Maria Wyrzykowska

ADER: Adaptively Distilled Exemplar Replay Towards Continual Learning for Session-based Recommendation

FEI MI, XIAOYU LIN, and BOI FALTINGS, Artificial Intelligence Laboratory, Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland

RecSys 2020

Session-based Recommendations

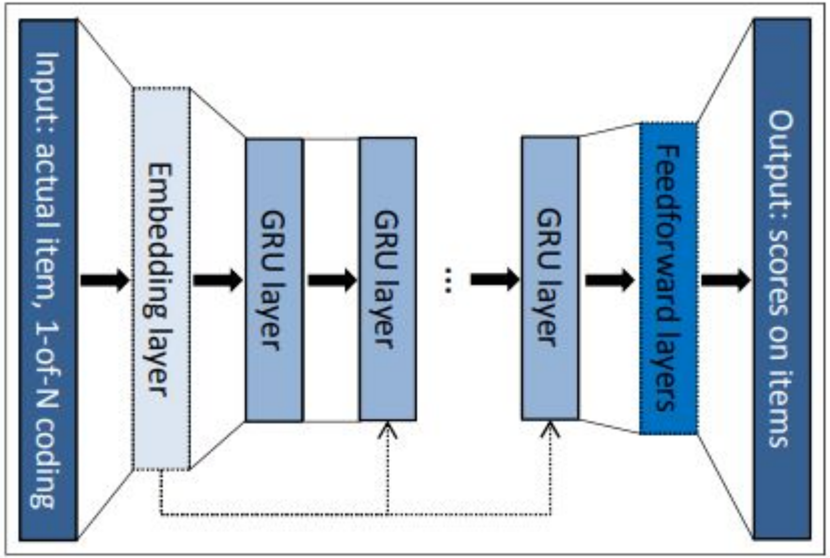
- users’ id is **unknown** (privacy reasons)
- only anonymous, short-term interaction **data within a browser session** is known
- **task**: based on sequence of actions (clicks, views), predict next one
- traditional MF methods are not very useful

Examples of models for session-based recommendations

Decoder/encoder **neural networks** or **KNN** methods are popular:

- Gru4Rec,
- STMP/STAMP
- SASRec
- SKNN

Gru4Rec: GRU-based RNN



- **traditional RNN** network with residual connections
- **input:** sequence of one-hot encodings of items
- **output:** scores on items
- training modified to suit the task

Gru4Rec: training modifications

1. **Sampling on the output**:
   - number of items is big; while training we compute the scores only for the positive item and a sample of negative items
   - sampling is done in proportion to popularity
2. **Loss functions**:
   - BPR (Bayesian Personalized Ranking):

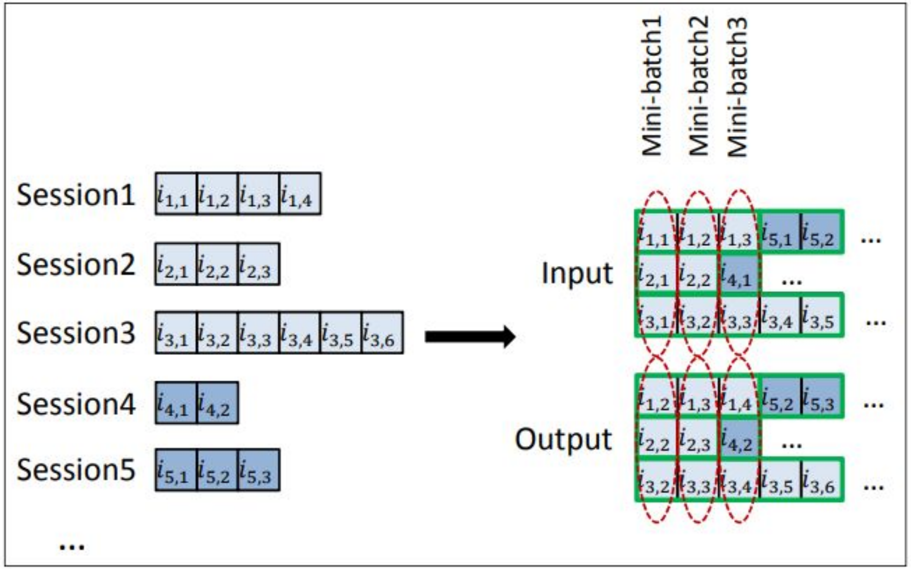
$$L_s = -\frac{1}{N_s}\sum_{j=1}^{Ns} log(\sigma(r_{s,i} - r_{s,j}))$$

   - TOP1 (approximation of positive item rank):

$$L_s = \frac{1}{N_s}\sum_{j=1}^{Ns} \sigma(r_{s,j} - r_{s,i}) + \sigma(r_{s,j}^2)2$$

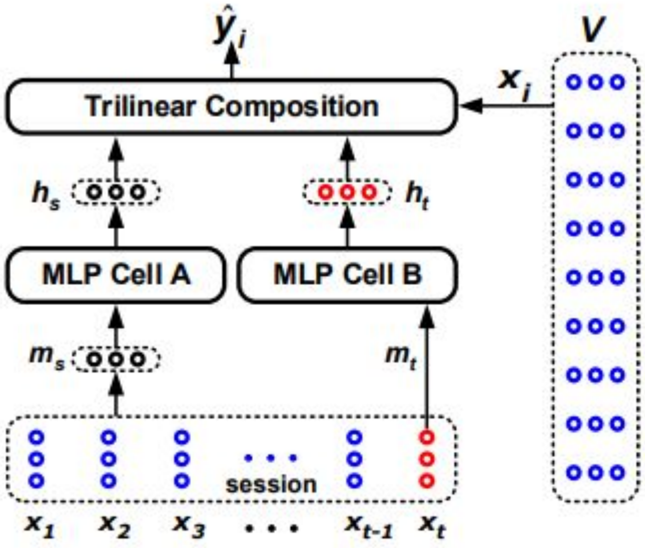
3. **Session-parallel mini-batches**

Gru4Rec: training modifications



- in NLP tasks, RNN usually use **in-sequence** mini-batches produced by sliding window; hidden state is reset after a batch
- **problems**: length of sessions varies a lot & we want to capture how sessions evolve over time -> session-parallel mini-batches
- hidden state is reset only when any of the session ends

STMP: Short-Term Memory Priority Model



- $x_{1,} ..., x_t$ - embedding of items in current session until timestamp t
- m_s - average of $x_{1,} ..., x_{t}$, $m_t = x_t$
- V - set of all items
- score for item x_i: y_i

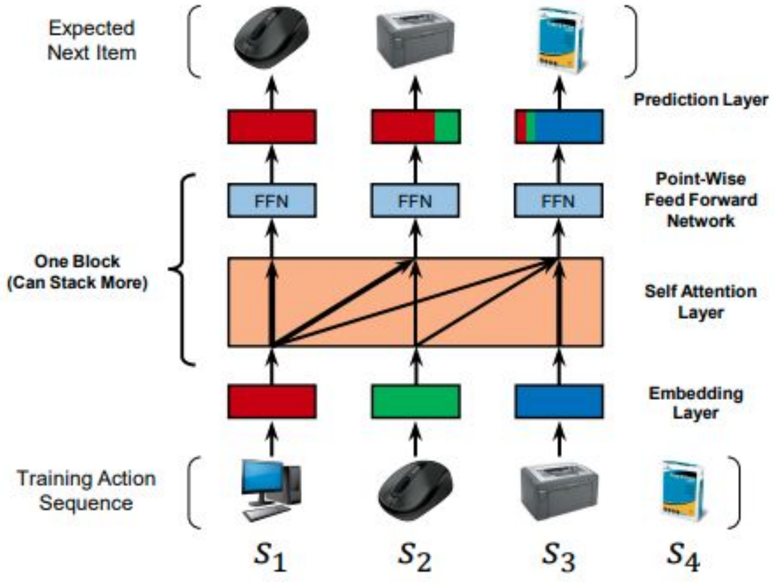
$$< a, b, c >= \sum_{i=1}^{d} a_i b_i c_i = \mathbf{a}^T (\mathbf{b} \odot \mathbf{c})$$

$$\hat{\mathbf{z}}_i = \sigma(< \mathbf{h}_s, \mathbf{h}_t, \mathbf{x}_i >)$$

$$\hat{\mathbf{y}} = softmax(\hat{\mathbf{z}})$$

- STAMP: using attention to produce m'_s based on $m_s + x_1, ..., x_t + x_t$

SASRec: Self-Attentive Sequential Recommendation



Architecture used in ADER paper.

Inspired by Transformer:

- input is embedded (including position in sequence)
- self-attention uses masked embeddings
- feed-forward network with non-linearity,
- MF prediction layer calculating relevance of items:

$$r_{i,t} = \mathbf{F}_t^{(b)} \mathbf{N}_i^T$$

SKNN: session-based kNN

Very basic approach, which can achieve results better than Gru4Rec.

Given session s (give as binary item) and its neighbours N_s, score for item i:

$$score_{\mathrm{SKNN}}(i, s) = \Sigma_{n \in N_s} sim(s, n) \cdot 1_n(i)$$

Sequence aware extensions:

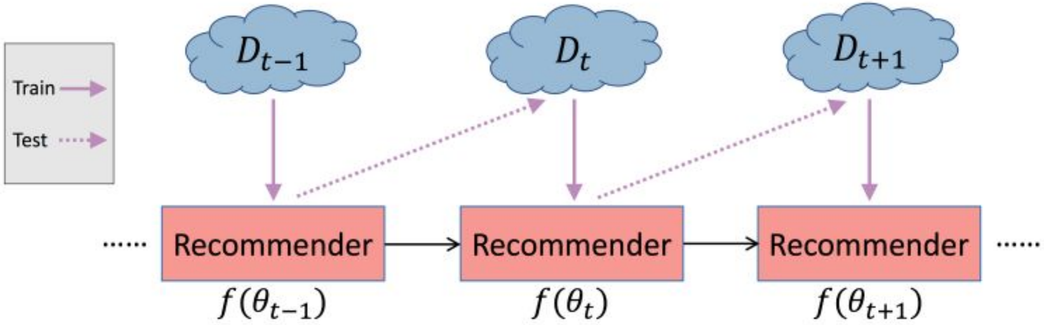
- encoding sessions as real-valued vectors and using dot product
- scoring function including weight (bigger when recent items match between s and n)

Back to ADER

= “Adaptively Distilled Exemplar Replay”
= continual learning setup

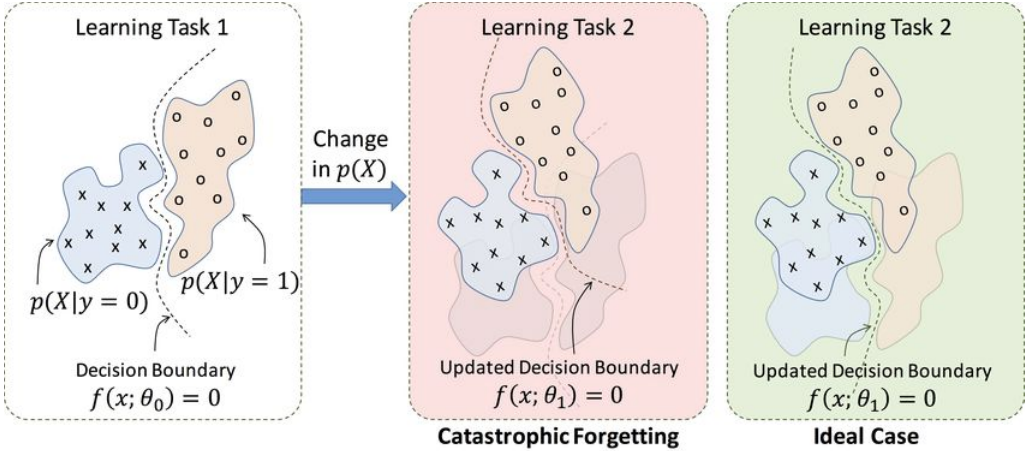
Continual learning

- offline training and evaluation of recommendation systems is unrealistic
- realistic setting: periodical updates of system with new data -> **continual learning**



Catastrophic forgetting

- if the model is trained only with new data at each timestep, it can forget what it had learned before
- **solutions:**
    - dynamic architectures
    - *exemplar replay*
    - *regularization*



Choosing the number of exemplars per item

$$m_{t,i} = N \cdot \frac{|\{\mathbf{x}, y = i\} \in D_t \cup E_{t-1}|}{|D_t \cup E_{t-1}|}$$

($\mathbf{x}$, y) - (sequence of items, target item)

D_t - training data from timestamp t

E_{t-1}- exemplars from timestamp t-1

N - number of exemplars in total

More popular item -> more exemplars

Choosing the exemplars

Algorithm 1 *ADER*: ExemplarSelection at cycle t

Input: $\mathcal{S} = D_t \cup E_{t-1}$; $M_t = [m_1, m_2, ..., m_{|I_t|}]$
  for $y = 1, ..., |I_t|$ **do**
    $\mathcal{P}_y \leftarrow \{\mathbf{x} : \forall(\mathbf{x}, y) \in \mathcal{S}\}$
    $\mu \leftarrow \frac{1}{|\mathcal{P}_y|} \sum_{\mathbf{x} \in \mathcal{P}_y} \phi(\mathbf{x})$
    for $k = 1, ..., m_y$ **do**
      $\mathbf{x}^k \leftarrow \arg\min_{\mathbf{x} \in \mathcal{P}_y} \|\mu - \frac{1}{k}[\phi(\mathbf{x}) + \sum_{j=1}^{k-1} \phi(\mathbf{x}^j)]\|$
    end for
    $E_y \leftarrow \{(\mathbf{x}^1, y), ..., (\mathbf{x}^{m_y}, y)\}$
  end for
Output: exemplar set $E_t = \cup_{y=1}^{|I_t|} E_y$

($\mathbf{x}$, y) - (sequence of items, target item)

D_t - training data from timestamp t

E_{t-1} - exemplars from timestamp t-1

M_t - vector storing number of exemplars per item

I_t - set of items

ϕ - encoder of session, e. g. neural network

We iteratively choose the exemplar which embedding best approximates the residual of average feature vector.

Knowledge distillation

If the number of exemplars is small, we need stronger constraint to not forget old patterns:

$$L_{KD}(\theta_t) = -\frac{1}{|E_{t-1}|} \sum_{(\mathbf{x}, y) \in E_{t-1}} \sum_{i=1}^{|I_{t-1}|} \hat{p}_i \cdot log(p_i)$$ = cross entropy

where: $[p_1, \ldots, p_{|I_{t-1}|}]$ are predicted probabilities over items generated by $f(\theta_t)$

$[\hat{p}_1, \ldots, \hat{p}_{|I_{t-1}|}]$ are predicted probabilities over items generated by $f(\theta_{t-1})$

Loss & training

Loss = cross entropy + knowledge distillation

$$L_{CE}(\theta_t) = -\frac{1}{|D_t|}\sum_{(\mathbf{x},y)\in D_t}\sum_{i=1}^{|I_t|}\delta_{i=y}\cdot log(p_i)$$

$$L_{ADER} = L_{CE} + \lambda_t \cdot L_{KD}, \quad \lambda_t = \lambda_{base}\sqrt{\frac{|I_{t-1}|}{|I_t|}\cdot\frac{|E_{t-1}|}{|D_t|}}$$

λ_t increases when there is less active items or number of exemplars increases.

Algorithm 2 *ADER*: UpdateModel at cycle t

Input: $D_t, E_{t-1}, I_t, I_{t-1}$
Initialize θ_t with θ_{t-1}
while θ_t not converged **do**
Train θ_t with loss in Eq. (4)
end while
Compute E_t using Algorithm 1 with θ_t and M_t computed by Eq. (1)
Output: updated θ_t and new exemplar set E_t

Experiments: datasets

- click-streams on e-commerce sites over 5-6 months
- DIGINETICA:
  - splitted by week
  - dynamic
- YOOCHOOSE:
  - splitted by day
  - more data
- 16 cycles

Experiments: results

	DIGINETICA					YOOCHOOSE				
	Finetune	*Dropout*	*EWC*	*Joint*	*ADER*	*Finetune*	*Dropout*	*EWC*	*Joint*	*ADER*
Recall@20	47.28%	49.07%	47.66%	50.03%	**50.21%**	71.86%	72.20%	71.91%	72.22%	**72.38%**
Recall@10	35.00%	36.53%	35.48%	37.27%	**37.52%**	63.82%	64.15%	63.89%	64.16%	**64.41%**
MRR@20	16.01%	16.86%	16.28%	17.31%	**17.32%**	36.49%	36.60%	36.53%	36.65%	**36.71%**
MRR@10	15.16%	16.00%	15.44%	16.43%	**16.45%**	35.92%	36.03%	35.97%	36.08%	**36.14%**

Model architecture: **SASRec**

MRR: mean reciprocal rank: $$\mathrm{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\mathrm{rank}_i}$$

Experiments: different number of exemplars

	10k	20k	30k
Recall@20	49.59%	50.05%	50.21%
Recall@10	36.92%	37.40%	37.52%
MRR@20	17.04%	17.29%	17.32%
MRR@10	16.17%	16.42%	16.45%

Experiments: ablation study

	ER_{random}	ER_{loss}	$ER_{herding}$	$ADER_{equal}$	$ADER_{fix}$	$ADER$
Recall@20	49.14%	49.31%	49.34%	49.92%	50.09%	50.21%
Recall@10	36.61%	36.65%	36.78%	37.21%	37.41%	37.52%
MRR@20	16.79%	16.90%	16.85%	17.23%	17.29%	17.32%
MRR@10	15.92%	16.02%	16.98%	16.35%	16.41%	16.45%

- ER_{random} - exemplars are selected at random
- ER_{loss} - exemplars with smallest cross-entropy are selected
- $ER_{herding}$ - no knowledge distillation in loss
- $ADER_{equal}$ - equal number of exemplars per item is selected
- $ADER_{fix}$ - λ_t in loss is fixed

Conclusion

- session-based recommendations look like interesting area of recommender systems, but most of the papers related to recommendations skip over real-life difficulties they pose
- ADER is an easy, model-agnostic method which could be useful in continual learning
- experimental setting looks pretty reliable, but achieved results are not groundbreaking

Thanks for your attention!

Sources

ADER: https://arxiv.org/pdf/2007.12000.pdf, https://github.com/doublemul/ADER

Gru4Rec: https://arxiv.org/pdf/1511.06939.pdf

STAMP: https://dl.acm.org/doi/pdf/10.1145/3219819.3219950

SASRec: https://arxiv.org/pdf/1808.09781.pdf

SKNN: https://arxiv.org/pdf/1803.09587.pdf