Recurrent Neural Networks with Top-k Gains for Session-based Recommendations

Balázs Hidasi
Gravity R&D
Budapest, Hungary
balazs.hidasi@gravityrd.com

Alexandros Karatzoglou
Telefonica Research
Barcelona, Spain
alexk@tid.es

Introduction

- **The exact ranking or scoring of items** in the tail of the item list (items that the user will not like) **is not that important**, but **it is very important to rank correctly** the items that the user will like at the top of the list (first 5, 10 or 20 positions). To achieve this with machine learning, one has to typically utilize **learning to rank techniques** and in particular ranking objectives and loss functions.
- **Pairwise loss** and **listwise loss** are typical loss functions of learning to rank. These loss functions work well in matrix factorization dose not guarantee in any way that they are optimal choice for RNNs as **backpropagation requirements** are stronger than those posed by simple SGD.

Sampling

- **Sampling** is essential and crucial for learning to rank task.
- When utilizing a sampling procedure, **it is important that high scoring items are included among the negative samples** to update gradients effectively.
- **Popularity-based sampling** is a good sampling strategy, since popular items generally score high in many situations.
- **Mini-batch sampling is basically a form of popularity-based sampling**, since the training iterates through all events, thus the probability of an item acting as a negative sample is proportional to its support.
- Since popularity-based sampling will focus more on popular ones, **learning can slow down** and can still be **inaccurate for ranking long tail high scoring items**.

Sampling

- Utilizing the parallelization capabilities of current hardware better, thus **training faster**, and **producing more stable gradients** than stochastic gradient training and thus converging faster.
- **Mini-batch sizes are generally small**, ranging from few tens to few hundreds. If the number of items is large, the small sample size further hinders the chance of including all of the high scoring negative examples.
- **There is a trade-off between accuracy and parallelization**. Small mini-batch size (30-100) produces more accurate models, but training with larger ones is faster on the GPU due to parallelization.

Loss Function

- **Categorical cross-entropy**

Categorical cross-entropy measures the distance of a proposed (discrete) probability distribution q from the target distribution p.

$$H(p, q) = -\sum_{j=1}^{N} p_j \log q_j$$

It is common practice to use the *softmax* to transform the output scores as a form a distribution.

$$s_i = \frac{e^{r_i}}{\sum_{j=1}^{N} e^{r_j}}$$

$$L_{\text{xe}} = -\log s_i = -\log \frac{e^{r_i}}{\sum_{j=1}^{N} e^{r_j}}$$

Loss Function

- Cross-entropy in itself is a **pointwise loss**, as it is the sum of independent losses defined over the coordinates.
- Assuming that there is a k for which $r_k \gg r_i$, s_i becomes very small and rounded to 0, because of the **limited precision**. The loss then computes log0, which is undefined and **instability**.
- Two ways to circumvent this problem are as follow: (a) compute $-\log(s_i + \epsilon)$, where ϵ is a very small value. (b) compute $-\log s_i$ directly as $-r_i + log \sum_{j=1}^{N} e^{r_j}$. The former introduces some noise, while the latter does not allow the separated use of the transformation and the loss, but both methods stabilize the loss and have any differences in the results.

Loss Function

- **Ranking losses: TOP1 & BPR**

$$L_{\text{top1}} = \frac{1}{N_S}\sum_{j=1}^{N_S}\sigma(r_j - r_i) + \sigma(r_j^2) \qquad \frac{\partial L_{\text{top1}}}{\partial r_i} = -\frac{1}{N_S}\sum_{j=1}^{N_S}\sigma(r_j - r_i)\left(1 - \sigma(r_j - r_i)\right)$$

$$L_{\text{bpr}} = -\frac{1}{N_S}\sum_{j=1}^{N_S}\log\sigma(r_i - r_j) \qquad \frac{\partial L_{\text{bpr}}}{\partial r_i} = -\frac{1}{N_S}\sum_{j=1}^{N_S}\left(1 - \sigma(r_i - r_j)\right)$$

With pairwise losses, one generally wants to have negative samples with high scores, as those samples produce high gradients and update the model. Denote the samples where $\boldsymbol{r_j \ll r_i}$ as irrelevant. **For this irrelevant sample $\boldsymbol{\sigma(r_j - r_i)}$ in $\boldsymbol{\partial_{L_{top1}}}$ and $\boldsymbol{1 - \sigma(r_i - r_j)}$ in $\boldsymbol{\partial_{L_{bpr}}}$ will be close to zero, which will cause gradients vanish and thus learning stops.** Meanwhile the gradient is always **discounted by the total number of negative samples**.

Loss Function

- **Ranking-max loss function family**

The idea is to have the target score compared with the most relevant sample score, which is the maximal score amongst the samples.

$$L_{\text{pairwise-max}}\left(r_i, \{r_j\}_{j=1}^{N_S}\right) = L_{\text{pairwise}}(r_i, \max_j r_j)$$

The maximum selection is non-differentiable and thus cannot be used with gradient descent. Therefore we use the *softmax* scores to preserve differentiability.

Loss Function

- **TOP1-max & BPR-max**

$$L_{\text{top1-max}} = \sum_{j=1}^{N_S} s_j \left(\sigma(r_j - r_i) + \sigma(r_j^2) \right) \qquad \frac{\partial L_{\text{top1-max}}}{\partial r_i} = -\sum_{j=1}^{N_S} s_j \sigma(r_j - r_i) \left(1 - \sigma(r_j - r_i)\right)$$

$$L_{\text{bpr-max}} = -\log \sum_{j=1}^{N_S} s_j \sigma(r_i - r_j) \qquad \frac{\partial L_{\text{bpr-max}}}{\partial r_k} = s_k - \frac{s_k \sigma^2(r_i - r_k)}{\sum_{j=1}^{N_S} s_j \sigma(r_i - r_j)}$$

The individual losses weighted by the corresponding *softmax* scores s_j.

$s_j \sigma(\cdot)$ can be regarded as the weight of loss function, which is able to select the relevant items (high gradients) to avoid gradients vanish.

Loss Function

- **BPR-max with score regularization**

For TOP1 loss, if only the first condition is met, the gradient w.r.t. r_i might vanish, but **the regularizing part of TOP1 makes sure that r_j is moved towards zero**, which might even make the update possible for r_i next time. The score regularization in TOP1 is very beneficial to the overall learning process, so even though the loss might not be theoretically optimal, it can achieve good results. Therefore we added score regularization to the BPR-max loss function as well.

$$L_{\text{bpr-max}} = -\log \sum_{j=1}^{N_S} s_j \sigma(r_i - r_j) + \lambda \sum_{j=1}^{N_S} s_j r_j^2$$

Experiments

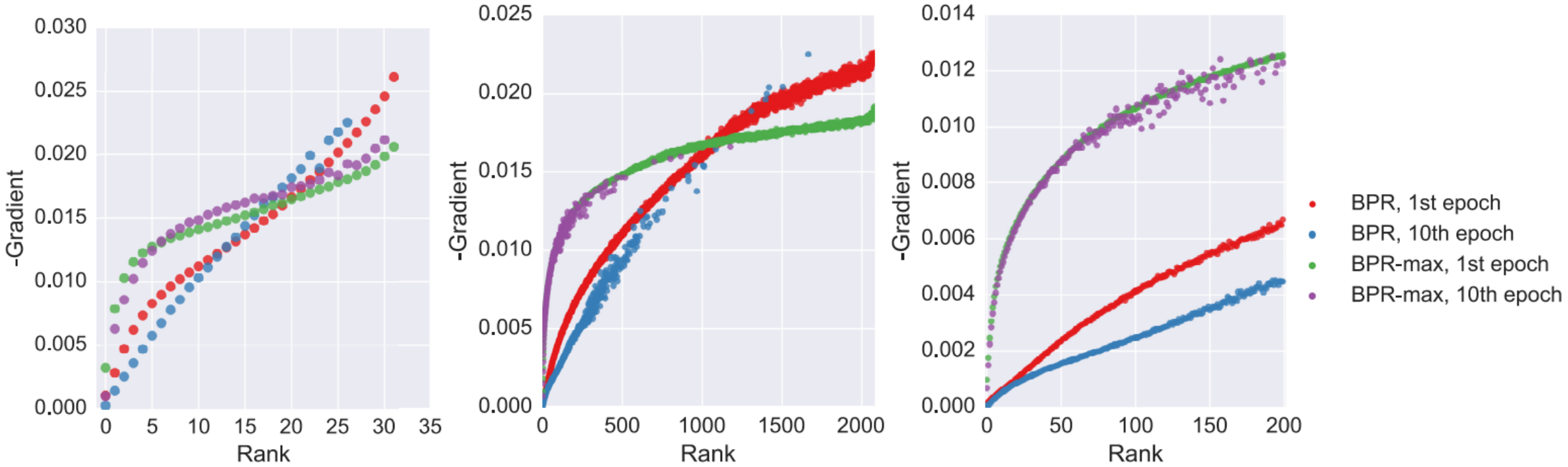

Figure 2: Median negative gradients of BPR and BPR-max w.r.t. the target score against the rank of the target item. Left: only minibatch samples are used (minibatch size: 32); Center: 2048 additional negative samples were added to the minibatch samples; Right: same setting as the center, focusing on ranks 0-200.

Experiments

Table 2: Recommendation accuracy with additional samples and different loss functions compared to item-kNN and the original GRU4Rec. Improvements over item-kNN and the original GRU4Rec (with TOP1 loss) results are shown in parentheses. Best results are typeset bold.

Dataset	Item kNN	GRU4Rec original	GRU4Rec XE	GRU4Rec with additional samples TOP1	XE	TOP1-max	BPR-max
				Recall@20			
RSC15	0.5065	0.5853	0.5781	0.6117 (+20.77%, +4.51%)	0.7208 (+42.31%, +23.15%)	0.7086 (+39.91%, +21.07%)	**0.7211 (+42.37%, +23.20%)**
VIDEO	0.5201	0.5051	0.5060	0.5325 (+2.40%, +5.43%)	0.6400 (+23.06%, +26.72%)	0.6421 (+23.46%, +27.12%)	**0.6517 (+25.31%, +29.03%)**
VIDXL	0.6263	0.6831	0.7046	0.6723 (+7.35%, -1.58%)	0.8028 (+28.19%, +17.53%)	0.7935 (+26.70%, +16.16%)	**0.8058 (+28.66%, +17.97%)**
CLASS	0.2201	0.2478	0.2545	0.2342 (+6.41%, -5.50%)	0.3137 (+42.54%, +26.61%)	0.3252 (+47.75%, +31.22%)	**0.3337 (+51.61%, +34.66%)**
				MRR@20			
RSC15	0.2048	0.2305	0.2375	0.2367 (+15.61%, +2.69%)	0.3137 (+53.16%, +36.08%)	0.3045 (+48.70%, +32.08%)	**0.3170 (+54.78%, +37.52%)**
VIDEO	0.2257	0.2359	0.2609	0.2295 (+1.69%, -2.73%)	0.3079 (+36.42%, +30.52%)	0.2950 (+30.72%, +25.05%)	**0.3089 (+36.87%, +30.95%)**
VIDXL	0.3740	0.3847	0.4343	0.3608 (-3.53%, -6.21%)	0.5031 (+34.52%, +30.78%)	0.4939 (+32.05%, +28.39%)	**0.5066 (+35.45%, +31.68%)**
CLASS	0.0799	0.0949	0.0995	0.0870 (+8.83%, -8.36%)	0.1167 (+46.08%, +22.99%)	0.1198 (+49.93%, +26.25%)	**0.1202 (+50.40%, +26.63%)**

Experiments

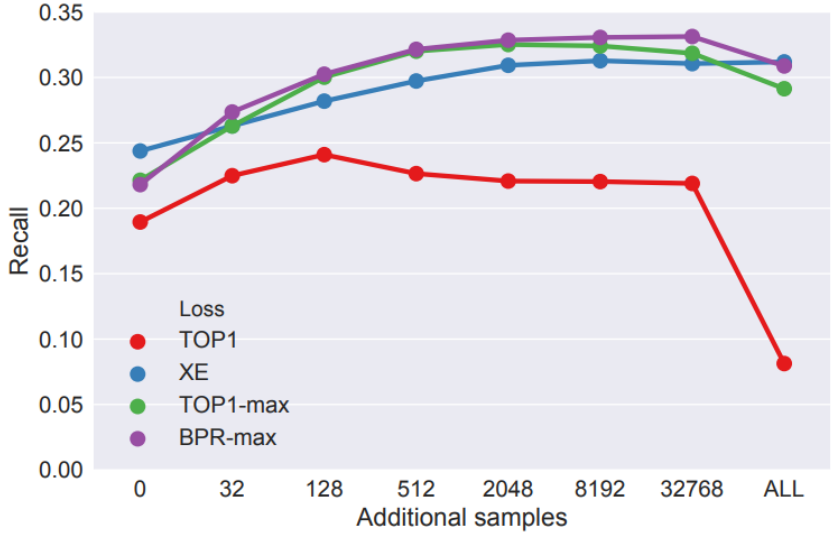

Figure 3: Recommendation accuracy with additional samples on the CLASS dataset. "ALL" means that there is no sampling, but scores are computed for all items in every step.

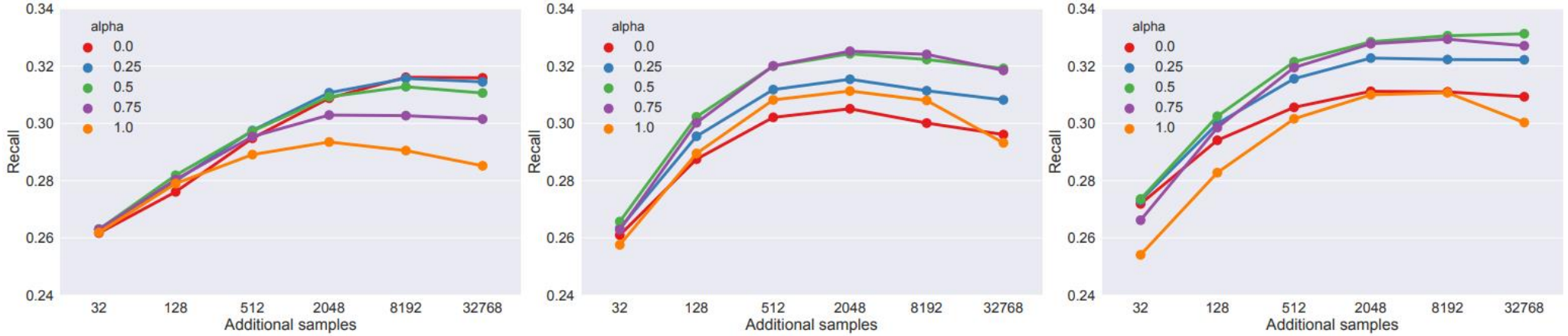

Figure 5: The effect of the alpha parameter on recommendation accuracy at different sample sizes on the CLASS dataset. Left: cross-entropy loss; Middle: TOP1-max loss; Right: BPR-max loss.