Learning to Rank For Push Notifications Using Pairwise Expected Regret

Yuguang Yue*, Yuanpu Xie, Huasen Wu, Haofeng Jia, Shaodan Zhai, Wenzhe Shi, Jonathan J Hunt*

Twitter Inc,

Email: {yuguangy, yxie, huasenw, hjia, szhai, wshi, jjh}@twitter.com

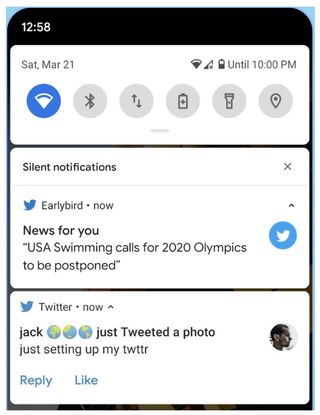
Summary

Push notifications have different properties than prior ranking problems. We introduced a novel *expected regret* (ER) ranking loss designed for push notifications, which weights a pairwise loss to minimize the expected regret in the ranking. We compared ER against prior approaches in both a simulation and a real-world production setting. In both cases, we found significant improvements in performance over prior methods.

Push notifications present new challenges for information retrieval and ranking. We hope to encourage further research in this topic.

More details on this work available at `https://arxiv.org/abs/2201.07681`

Push Notification ranker at Twitter



Push notification differs from classical ranking problems in several key ways:

- The user will observe only one notification from each set of candidates.
- User behavior is highly personalized since there is no explicit context from the user to indicate their information need.
- User responses are highly non-stationary, a notification that is timely and relevant now, may be irrelevant if sent later
- The candidates available to be sent to the user change rapidly and are highly personalized, so the approach must generalize to rank documents never seen before.

Pairwise loss with expected regret weights

For a given user u, assume we have features x and labels y for all Tweets in a candidate set (the set of Tweets we choose from for push notification) and we know the true click through rate (CTR) $\tilde{y} = p(y|x, u)$ for those Tweets.

We weight the loss between each positive and negative pair (x_{pos}, x_{neg}) by the expected regret of misordering these two Tweets when the positive Tweet should have been sent. More specifically:

$$w'_{er} = \underbrace{p_{top}(\tilde{y}_{pos})}_{\text{probability } x_{pos} \text{ is the optimal Tweet}} \times \underbrace{(\tilde{y}_{pos} - \tilde{y}_{neg})}_{\text{regret of CTR if } x_{neg} \text{ Tweet is sent instead}}$$

$$w_{er} = \underbrace{\max(w'_{er}, k)}_{\text{prevent weights from degenerating}}$$

where the probability that a candidate with a CTR of $\tilde{y}_{pos}$ is the top ranked candidate in a candidate set of size n is:

$$p_{top}(\tilde{y}_{pos}) = (1 - F(\tilde{y}_{pos}))^{n-1}$$

The expected regret loss over a candidate set for user u can be defined as

$$\ell_{er}(u, \theta) = \sum_{\mathcal{X}_{pos}} \sum_{\mathcal{X}_{neg}} w_{er}(x_{pos}, x_{neg}) \times \max(0, 1 - (f_\theta(u, x_{pos}) - f_\theta(u, x_{neg})))$$

where $\mathcal{X}_{pos}$ and $\mathcal{X}_{neg}$ are the positive Tweets and negative Tweets in one candidate set.

Removing the assumptions

The assumptions in red are unrealistic to know in real world, and must be estimated in real applications.

1. We estimate the true CTR of a Tweet by adding binary classification loss

$$\ell_{ce}(u, \theta) = -\sum_{x_{pos} \in \mathcal{X}_{pos}} log(\underbrace{\sigma}_{\text{sigmoid function}}(f_\theta(u, x_{pos})) - \sum_{x_{neg} \in \mathcal{X}_{neg}} \log(1 - \sigma(f_\theta(u, x_{neg}))$$

to our final loss function and using the current model estimate for computing w_{er}.

2. We construct a pseudo candidate set by aggregating Tweet sents to different users of the same "user type" (a feature designed at Twitter that groups users based on their behavior). We do this grouping within each minibatch of data.

Pseudo Code

Initialize θ
for training iterations **do**
  Sample pair of examples $(u_{pos}, x_{pos}), (u_{neg}, x_{neg})$
  Estimate CTRs $\tilde{y}'_{pos} = f_\theta(u_{pos}, x_{pos}),\ \tilde{y}'_{neg} = f_\theta(u_{neg}, x_{neg})$.
  Compute w_{er} and the estimated values $\tilde{y}'_{pos}, \tilde{y}'_{neg}$.
  Compute ER loss ℓ_{er}
  Sample pointwise sample u_i, x_i, y_i.
  Compute pointwise loss ℓ_2.
  Update θ to minimize loss

$$\ell = \ell_{er} + \alpha\ell_2 \quad (1)$$

end for

Empirical results

We first run the ER model on a simulated dataset. We used the recsim framework to construct the simulation and we fit key parameters of the simulation to production data in order to make the simulation more realistic.

The simulated data is generated by first sampling the 'user type' from a Categorical distribution that fits the production distribution. Conditioned on the user type, a candidate set consisting of 60 documents is sampled at each interaction. Each document has an underlying probability of being opened $\tilde{y} = p(y = 1|x, u)$, which is drawn independently and identically from a Beta distribution fit (using maximum likelihood estimation) to the candidate distribution of the production system. Each document has a corresponding set of features x, which in our case is a 5 degree projection of $p(y = 1|x, u)$ with independent Gaussian noise added to each dimension.

A model is used to score the documents and select which one is sent to the user, and a label is generated by sampling from the Bernoulli distribution defined by the latent CTR $\tilde{y}$ associated with the document. For evaluation, we used the latent probability of open to compute the regret without any noise due to the label sampling.

Model	unbiased	gain
Pointwise	0.07624 ± 0.00006	0.0%
Pairwise	0.07623 ± 0.00006	0.01%
K-OS-AUC	0.07679 ± 0.00016	-0.72%
Expect Regret	$\mathbf{0.07602 \pm 0.00003}$	0.28%

Model	biased	gain
Pointwise	0.07417 ± 0.00005	0.0%
Pairwise	$\mathbf{0.07412 \pm 0.00004}$	0.07%
K-OS-AUC	0.08139 ± 0.00119	-9.73%
Expect Regret	0.07429 ± 0.00008	-0.16%

note: biased data represents data sampled from a pointwise ranker; unbiased data represents data sampled from a random ranker.

We launched an online experiment on Twitter's experimentation platform, where we compare our expected regret model with pointwise loss model and pairwise loss model and report key metrics here.

Model	Open	Interact
Pointwise	0.0%	0.0%
Pairwise	0.04%	0.46%
Expected regret	**0.33%**	**2.46%**