ABSTRACT. We present a web scale recommender system that has a small memory footprint, fast training time, and low prediction latency. We show that by using stochastic optimization with only a single pass over the data and sparsity constraints, we can efficiently learn ranking functions without sacrificing accuracy.

The picture below shows the "Customers Who Bought This Item Also Bought" widget shown on all product pages in AMAZON.com. The goal of this widget is to provide accurate recommendations and help users find relevant items. In this study, we are interested in using purchase logs to learn more accurate ranking models and improve the quality of product recommendations.

A variety of deployment constraints make the applicability of machine learning algorithms challenging:

1. The scale of the data: large number of examples, large number of features
2. Constant change in customer’s preferences: requires constant retraining
3. Making predictions in real time: to provide better customer experience

We address these challenges by making the following design decisions:
1. Stochastic optimization
2. Single-pass learning
3. Exploiting sparsity

• Ranking metrics (like NDCG) are hard to optimize explicitly, since they are discontinuous.
• LambdaRank [1] estimates the model parameters while implicitly accounting for the ranking metric.
• Parameter updates are weighted proportionally to changes in the ranking metric.

For a given ranking evaluation metric $\Delta(r, l)$, we introduce delta function:

$$\Delta_{i,j}(i,j) = M_i(r,i) - M_j(r,j)$$

Then, let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be the features associated to the products and $\varphi(x_i)$ be a parametric function. We define the ranking loss as follows:

$$\mathcal{L}(\mathcal{X}, \varphi) = \sum_{i,j} \Delta_{i,j}(i,j) \cdot \mathcal{P}(\varphi(x_i), \varphi(x_j))$$

Possible choices of $\varphi(x_i)$ include:

- $\mathcal{P}(x, y) = \max(0, y - x + e)$
- $\mathcal{P}(x, y) = \log(1 + \exp(y - x))$

If $\varphi(x_i) > \varphi(x_j)$, $\Rightarrow$ little loss is incurred
If $\varphi(x_i) < \varphi(x_j)$, $\Rightarrow$ the pairwise loss is weighted by $\Delta_{i,j}(i,j)$

To handle large high-dimensional datasets it is crucial to exploit sparsity and perform only minimum number of operations per update.

We add:

1. $L_1$-regularization term to enforce sparsity,
2. $L_2$-regularization term to improve convergence and add strong convexity.

$$\mathcal{L}_{\ell_1}(\mathcal{X}, \varphi) = \ell_{\ell_1}(\mathcal{X}, \varphi) + \lambda_1 ||w||_1 + \frac{1}{2} \lambda_2 ||w||^2_2$$

We consider three optimization algorithms for which an update for a single instance is linear in the number of non-zero features.

FOBOS: Forward-Backward Splitting [2], solves the regularized optimization problem by alternating between two phases:

1. Taking a simple gradient step:
   $$w_{t+1} = w_t - \eta_t \frac{\partial}{\partial w_t} \ell_{\ell_1}(\mathcal{X}, \varphi)$$

2. Taking a proximal step that involves the elastic-net regularization:
   $$w_{t+1} = \begin{cases} 0 & \text{if } |w_i| \leq \eta_t \lambda_1, \\ \frac{1}{1 + \eta_t\lambda_1} (w_i - \text{sgn}(w_i)\eta_t\lambda_1) & \text{otherwise}. \end{cases}$$

where $\eta_t$ is learning rate.

RDA: Regularized Dual Averaging [3], solves a proximal step involving the exponential weighted average is first updated as:

$$\overline{w}_{t+1} = \frac{1}{t+1} w_{t+1} + \frac{t}{t+1} \overline{w}_t$$

The weights are adjusted as follows:

$$w_{t+1} = \begin{cases} 0 & \text{if } |w_i| \leq \lambda_1, \\ \frac{1}{1 + \lambda_1\eta_t} (w_i - \text{sgn}(w_i)\lambda_1) & \text{otherwise}. \end{cases}$$

pSGD: Pruned Stochastic Gradient Descent, enforces sparsity by simply adding a pruning operation to the $L_2$-regularized SGD update rule.

Every $k$ steps we set:

$$|w_i| < \theta \Rightarrow w_i = 0$$

Goal: Assess how ElasticRank compares to state-of-the-art ranking models

Dataset: LETOR 3.0 benchmark collection

Next, we evaluate ElasticRank on a dataset collected on the Amazon retail website. To run the experiments, we sampled a set of impression log data from a contiguous time interval, and used 9/11 of the data for training and 1/11 for validation and testing, leaving the temporal order intact.

Next, we evaluate ElasticRank on a dataset collected on the Amazon retail website. To run the experiments, we sampled a set of impression log data from a contiguous time interval, and used 9/11 of the data for training and 1/11 for validation and testing, leaving the temporal order intact.

Next, we evaluate ElasticRank on a dataset collected on the Amazon retail website. To run the experiments, we sampled a set of impression log data from a contiguous time interval, and used 9/11 of the data for training and 1/11 for validation and testing, leaving the temporal order intact.

References