Heterogeneous Dyadic Multi-Task Learning with Implicit Feedback

Simon Moura[1], Amir Asarbaev[1,4], Massih-Reza Amini[1], and Yury Maximov[2,3]

[1]Univ. Grenoble Alps, CNRS, Grenoble INP - LIG, France
{*firstname.surname*}*@univ-grenoble-alpes.fr*
[2]Skolkovo Institute of Science and Technology, Russia
[3]Theoretical Division T-5 & CNLS Los Alamos National Laboratory, USA
y.maximov@skoltech.ru
[4]Moscow Institute of Physics and Technology, Russia
amir.asarbaev@grenoble-inp.org

Abstract. In this paper we present a framework for learning models for Recommender Systems (RS) in the case where there are multiple implicit feedback associated to items. Based on a set of features, representing the dyads of users and items extracted from an implicit feedback collection, we propose a stochastic gradient descent algorithm that learn jointly classification, ranking and embeddings for users and items. Our experimental results on a subset of the collection used in the RecSys 2016 challenge for job recommendation show the effectiveness of our approach with respect to single task approaches and paves the way for future work in jointly learning models for multiple implicit feedback for RS.

Keywords: Recommendation Systems, Multiple Implicit Feedback, Dyadic Prediction, Muti-task Learning

1 Introduction

The aim of Recommender Systems (RS) is to present products to users by adapting the displayed offers to their taste. Recently, there was a surge of interest in the design of efficient RS especially after the NetFlix challenge [1]; and also because of many new problems such as the study of an accurate and scalable RS presents. As most of the users interactions are now provided in the form of clicks, an active line of research on RS is to learn models based on implicit feedback. Although there is no evidence of the actual value of such feedback, as a positive (respectively negative) feedback on an item does not necessarily represents a user preference (respectively dislike), almost all approaches assume that positive feedback conveys relevant information for the problem at hand.

In this work, we consider the case where there are multiple implicit feedbacks for each items and propose a multi-target learning algorithm that enhance prediction over each of these feedbacks. We cast the problem as a dyadic prediction problem, where the aim is to predict multiple outputs for observations that are

constituted by pairs of examples formed by users and items. A classical approach when dealing with multiple outputs is to divide the problems into *simpler* sub-prediction problems and deal with them separately without considering their relationships. However, the intuition and the consensus is that, when some tasks are interdependent and potentially heterogeneous (that is, for instance, when some tasks deal with classification while others deal with ranking or regression), the learner will benefit from learning them jointly by taking into account the shared information. Following this intuition, multi-task (MTL) approaches[1] [2, 3] consider the first example of the dyad, an *observation*, and the second example, a *task*, and propose to solve the general prediction problem by taking advantage of the correlation between the tasks [4–7]. Most of these approaches learn a different model for each task using the feature representation of observation and model the dependencies in the objective function using a shared regularization term that enforces correlated tasks to have close models [4, 5, 8]. However, by simultaneously taking into account both instances of a dyad, one can expect to do better by building a single model and by using all the available information at once.

Contributions. Although, dyadic prediction problems are common [9], the case where they are associated with multiple and heterogeneous outputs is rare and still not fully studied mainly due to the lack of data collections. In this paper, we propose a generic method to extract a meaningful representation from multiple implicit feedback data for RS. Based on this method, we provide a dataset built over the RecSys 2016 competition for job recommendation. The competition consisted in proposing job offers to users that would be of their interest, with the particularity that users may have simultaneously *clicked*, *bookmarked*, *replied*, and *removed* specific offers that they have been proposed. We adapted the method of [10] that was initially designed for explicit feedback single task learning to the case of multiple implicit feedbacks. To evaluate the user-offer dyadic representations and analyze the usefulness of taking into account the relationship between the tasks (different implicit feedback), we propose a MTL stochastic gradient descent (SGD) algorithm that combines classification and ranking predictors and the learning of user and item embeddings. We show that the combination of tasks allows to considerably enhance the prediction of most of the tasks, particularly the predictions for the clicks. Empirical comparisons of the MTL approach with single task approach that considers each of the implicit feedback independently shows the efficiency of the proposed strategy.

Organization of the paper. In Section 2 we briefly present the RecSys 2016 challenge dataset and the features that were extracted from the implicit feedback. In Section 3 we describe the MTL learning framework considered throughout this paper. In Section 4, we describe the gradient descent for jointly learning multiple heterogeneous tasks and embeddings. In Section 5 we present experimental results that evaluate the effect of taking into account the dependency among the outputs. Finally, in Section 6, we discuss the outcome of this study and give some pointers for further research.

[1] `https://www.ngdata.com/icml-2013-tutorial-multi-target-prediction/`

2 Feature Extraction based on Implicit Feedback

In this Section, we briefly describe the RecSys 2016 challenge and present the steps we followed to extract the dataset as well as the proposed learning strategy for combining the outputs[2].

The RecSys 2016 challenge[3], hosted by the XING social network platform, was defined as a ranking problem of clicks for job offers. For this competition, four main files were made available, each containing information on users and jobs offers. The offers that were displayed to the users are called *impressions*. Each offer displayed could be interacted in four different manners, by clicking, bookmarking, replying or deleting the offer.

In this collection, users provided implicit feedback and may have different interactions with the same offer. In particular, a user could have clicked and also replied to the same offer. We used both of these information to extract characteristics for the pairs of users and items. We did not consider the information concerning deleted offers as it does not help to decide whether a user is willing to interact positively with an offer or not, which is the primary goal.

The statistics of the original interactions and impressions are summarized in the left column of Table 1. In this table, the *sparsity* represents the user-offer pairs for which there is no interaction at all. Figure 1 shows the number of users that have positive interactions with respects to the clicks, bookmarks and replies. Table 2 shows the number of offers displayed to users, as well as the number of suggestions that clicked, bookmarked and replied. As expected, the vast majority of users make very few interactions while few users interact a lot. It comes out that the offers that are bookmarked and replied, represent less than 5% of those that are displayed to the users, making both associated tasks challenging.

To extract relevant features, we relied on the approach described in [10], which is a method that uses neighbors preferences and retrieves statistics about the closest users interactions (regarding a predefined similarity). The main idea is to compute statistics that describe the net preference of a user for a given offer. The dyadic representation for a user-offer pair is hence composed of statistics summarized in 15 features to which is added 2 biases.

Due to a significant amount of data available and the substantial sparsity, we subsampled the dataset to keep only the users for which we had enough information to extract meaningful statistics. We decided to keep users with more than 30 interactions and offers which had been interacted with at least 30 times. Following this process, we obtained **819.226** dyadic user-offer pairs that we randomly split into training (30% of the original dataset) and test (70% of the original dataset) collections. The right column of Table 1 contains statistics describing the dataset that we obtain after using the subsampling method described above.

The rational behind the subsampling of the original dataset is twofold. First, the users who did not interact enough are not relevant to learn a predictive model

[2] We make available the extracted dataset as well as the codes for research purpose.
[3] `https://recsys.acm.org/recsys16/challenge/`

	RecSys'16	DAEMON
# Users	**770.859**	**5.949**
# Offers	**1.002.161**	**25.184**
Av. nb interactions/user	7	137.71
Med. nb interactions/user	3	120
Av. nb users/offer	5	32.53
Med. nb users/offer	2	12
Av. nb displayed/user	76	115.04
Av. nb clicked/user	7	42.19
Av. nb bookmarked/user	0.27	1.38
Av. nb replied/user	0.42	3.40
Sparsity	99.991%	99.45%

Table 1: Statistics over the *original* RecSys 2016 (left) and DAEMON (right) datasets. In the latter we only consider users which had at least 30 interactions and the offers which had been interacted at least 30 times. (*Med.* stands for median, *Av.* for average).

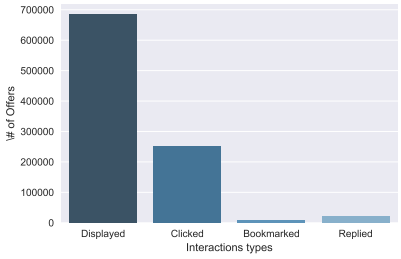

Table 2: Number of offers *displayed*, *clicked*, *bookmarked* and *replied*. *All* is the sum of the three latter. The dataset is highly imbalanced and contains only few *Bookmarked* and *Replied* interactions.

as we lack information about them. Secondly, the method we use to extract features [10] heavily relies on the similarity between users, which is less likely to be high in a sparse dataset.

To compute the statistics of pairs $(user, item)$, as proposed in [10], we associated to clicks, bookmarks and replies respectively the weights 1, 2 and 3, emphasizing the fact the replying to an offer shows more interest than simply clicking or bookmarking it.

Below we detail the exact steps followed to create representations for multiple implicit feedbacks:

1. Create a matrix (IM) of implicit interactions of users and offers summing over all positive interactions:

$$IM_{(u,o)} = \sum_{i \in \mathcal{I}} i * \mathbb{1}_{(u,o)_i = 1} \,,$$

   where $(u,o)_i = 1$ means that user u interacted positively with offer o, $\mathcal{I}$ represents the set of all possible interactions and $i = 1, 2$ or 3 for respectively clicked, bookmarked and replied interactions;
2. Evaluate users similarities (e.g. using cosine distance) using IM. Keep top K closest users for each users;
3. For each dyad (u, o) for which we have a ground truth (at least one positive interaction), compute the *win/tie/loss* vectors based on similar users than u that interacted with offer o as described in [10].
4. Finally, the features associated to the dyad (u, o) is obtained by considering 5 statistics over each of the *win/tie/loss* vectors, namely: the mean, the standard deviation, the max, the min and the normalized number of respectively wins, ties or losses.

3 Learning by Combining the Outputs

This section provides a formal definition of the multi-target learning framework considered in this article. We suppose that dyads are represented in an input space $\mathcal{X} \subseteq \mathbb{R}^d$ and that the output space $\mathcal{Y} = \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_T$ is a product of T different output spaces corresponding each to an interaction. Further, we assume that the pairs and their associated output $(\mathbf{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$ are generated i.i.d with respect to a fixed yet unknown joint probability distribution $\boldsymbol{\mathcal{D}} = (\mathcal{D}_1, \ldots, \mathcal{D}_T)$. The aim of learning is to find a prediction function in some predefined function set $\mathcal{H} = \{\boldsymbol{h} : \mathcal{X} \to \mathcal{Y}\}$, that minimizes the expected risk

$$\mathcal{L}(\boldsymbol{h}) = \mathbb{E}_{(\mathbf{x},\boldsymbol{y})\sim\boldsymbol{\mathcal{D}}}[\boldsymbol{\ell}(\boldsymbol{h}(\mathbf{x}), \boldsymbol{y})]\,, \tag{1}$$

where $\boldsymbol{\ell}(\boldsymbol{h}(\mathbf{x}), \boldsymbol{y})$ is an instantaneous loss measuring the discrepancy of the predictions over different outputs $\boldsymbol{h}(\mathbf{x}) = (\boldsymbol{h}_1(\mathbf{x}), \ldots, \boldsymbol{h}_T(\mathbf{x})) \in \mathcal{Y}$ of observation $\mathbf{x}$ and its desired output $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T) \in \mathcal{Y}$. Following the Empirical Risk Minimization principle, we achieve this aim by minimizing an empirical loss over a training set $\mathcal{S} = (\mathbf{x}_i, \boldsymbol{y}_i)_{i=1}^m$ of size m, where examples are be generated i.i.d with respect to the same probability distribution $\boldsymbol{\mathcal{D}}$.

In the case where we consider multiple tasks, the general formulation of the empirical loss function on a train set $\mathcal{S}$ can be written as follow:

$$\mathcal{L}_m(\boldsymbol{h}, \mathcal{S}) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{T}\sum_{j=1}^{T}\boldsymbol{\ell}_j(\boldsymbol{h}_j(\mathbf{x}_i), \boldsymbol{y}_i) + \lambda\Omega(\boldsymbol{h})\,, \tag{2}$$

where $\boldsymbol{\ell}_j$ is the loss for task j, $\boldsymbol{h}_j$ is the hypothesis function for tasks j and $\Omega(\boldsymbol{h})$ is a regularization term on the parameters of the models. In this work we aim at minimizing multiple loss functions over all models $\{\boldsymbol{h}_j\}_{j\in\{1,\ldots,T\}}$.

We considered two different setups to achieve this goal:

- Single task learning (STL) where the goal is to learn each prediction function $(h_j)_{1\leq j\leq T}$ independently of the others by minimizing the associated empirical loss;
- Multi-target learning (MTL) where the goal is to learn all prediction functions jointly by taking into account dependencies between their outputs. While the classical way of binding models is to use a shared regularization across the tasks, in our approach, the multi-task is carried out by a shared representation, that is also learned during the same training phase.

4 SGD for Multi-Target Heterogeneous Dyadic Learning

Based on the dataset extracted following the steps described in Section 2, we defined a set of relevant tasks for RS. As the main source of revenue for many online website rely on the number of clicks, the main goal remains to improve the predictions for the clicks. In this sense, we define the following tasks:

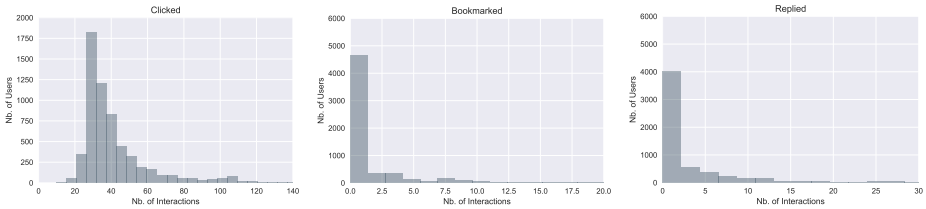

Fig. 1: The number of users with respect to the number of interactions. From left to right, for *Clicked*, *Bookmarked* and *Replied* tasks.

1. A *classical* learning to rank task in RS for clicking interactions, where the goal is to provide a ranking list of offers on which a user is willing to click. As explained earlier, the main source of revenues of many website is based on the number of clicks they register. In this scenario, it makes sense to "specialize" the models for the predictions of clicks. We will discuss more about this question and how we do that in practice in the definition of the fourth task.
2. Two binary classification tasks, where the goal is to predict if a user is willing to interact positively with specific offers in terms of bookmark and reply interactions.
3. A representation learning task, where the goal is to learn a meaningful representation $\mathcal{U}$ for users and $\mathcal{I}$ for items. This tasks is used as a binding between of the different models.

Based on the definition of the four tasks described above, we can formally re-write the loss functions for the tasks at hand as:

$$\begin{aligned}\mathcal{L}_m(\boldsymbol{h}, S) =& \mathcal{L}^{click}_{rank}(\boldsymbol{h}_1, \mathcal{S}) + \mathcal{L}^{book}_{class}(\boldsymbol{h}_2, \mathcal{S}) + \mathcal{L}^{reply}_{class}(\boldsymbol{h}_3, \mathcal{S}) + \mathcal{L}^{emb}_{rank}(\mathbb{U}, \mathbb{I}, \mathcal{S}) + \\ & \lambda \left(||\boldsymbol{h}_1||_2^2 + ||\boldsymbol{h}_2||_2^2 + ||\boldsymbol{h}_3||_2^2 + ||\mathbb{U}||_2^2 + ||\mathbb{I}||_2^2 \right) . \end{aligned} \tag{3}$$

We define each tasks as the average of all logistic losses evaluated for each example in the sample. In the case of pairwise ranking we note as $\mathcal{U} \subseteq N$ (resp. $\mathcal{I} \subseteq N$) the set of indexes over users (resp. the set of indexes over items). Furthermore, for each user $u \in \mathcal{U}$, we consider two subsets of offers, the offers interacted negatively $\mathcal{I}_u^- \subset \mathcal{I}$ (the user did not click) and the offers interacted positively $\mathcal{I}_u^+ \subset \mathcal{I}$ (the user clicked) such that:

- $\mathcal{I}_u^- \neq \emptyset$ and $\mathcal{I}_u^+ \neq \emptyset$.
- For any pair of offers $(i^+, i^-) \in \mathcal{I}_u^+ \times \mathcal{I}_u^-$, $i^+ \underset{u}{\succ} i^-$ mean that user u has a preference for item i^+ over item i^-.

Based on this preference relation, the ranking output $y_{i^+,u,i^-} \in \{-1, +1\}$ is defined over a triplet $(i^+, u, i^-) \in \mathcal{I}_u^- \times U \times \mathcal{I}_u^-$ as:

$$y_{i^+,u,i^-} = \begin{cases} +1 & \text{if } i^+ \underset{u}{\succ} i^- \\ -1 & \text{otherwise.} \end{cases} \tag{4}$$

In the case of the ranking problem for the clicks, $\boldsymbol{h}_1$ is of the form

$$\boldsymbol{h}_1(\mathbf{x}) = \langle \mathbf{w}_1, \mathbf{x} \rangle \,,$$

and we say that the model $\mathbf{w}_1$ is making an error on a prediction when it ranks higher a negative example over a positive example. Thus, we can compute the error made on one triplet $(i^+, u, i^-) \in \mathcal{I}_u^- \times U \times \mathcal{I}_u^-$ as

$$\mathbb{1}_{\langle \mathbf{w}_1, (u,i^+) \rangle < \langle \mathbf{w}_1, (u,i^-) \rangle} = \mathbb{1}_{\langle \mathbf{w}_1, (u,i^+) - (u,i^-) \rangle < 0} \,. \tag{5}$$

The loss function of Equation 5 is hard to optimize but can be approximated by a smooth surrogate function, such as the logistic loss function. We can re-write the loss for the ranking problem over a training set $\mathcal{S}$ as follow

$$\begin{aligned}
\mathcal{L}_{rank}^{click}(\mathbf{w}_1, \mathcal{S}) &= \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{(i^+,i^-) \in \mathcal{I}_u^+ \cup \mathcal{I}_u^-} \mathbb{1}_{\langle \mathbf{w}_1, (u,i^+) \rangle < \langle \mathbf{w}_1, (u,i^-) \rangle} \\
&\approx \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{(i^+,i^-) \in \mathcal{I}_u^+ \cup \mathcal{I}_u^-} \log(1 + e^{-y_{i^+,u,i^-} \langle \mathbf{w}_1, (u,i^+) - (u,i^-) \rangle}) + \lambda_1 ||\mathbf{w}_1||_2^2 \,.
\end{aligned}$$

Remark 1. Note that, for a fixed user, it is possible to have a different polarity of interactions with respect to clicks, bookmarks and replies. In other words, as we select pairs of offers (i^+, i^-) regarding the clicks polarity, it is possible to have i^+ and i^- both positives, or negatives, for bookmarks or replies.

In terms of binary classification with outputs $\boldsymbol{y}_i \in \{-1, +1\}$, a model $\boldsymbol{h}_2$ makes an error when its prediction differs from the ground truth $\boldsymbol{y}$. We can write the zero-one loss for classification as

$$\mathbb{1}_{\boldsymbol{y}_i \boldsymbol{h}_2(\mathbf{x}) < 0}, \text{ where } \boldsymbol{h}_2(\mathbf{x}) = \langle \mathbf{w}_2, \mathbf{x} \rangle \,.$$

Similar to the case of ranking described above, we use a logistic loss in order to approximate the zero-one loss for the two classification tasks at hand. We also average the loss over the two pairs. Note that the polarity can be different for the bookmarks and replies tasks, we will note $\boldsymbol{y}_{i^+}^b$ (respectively $\boldsymbol{y}_{i^-}^b$) the polarity for the positives (respectively negatives) interactions for the bookmark task for user $u \in \mathcal{U}$ and offers $(i^+, i^-) \in \mathcal{I}^2$ and $\boldsymbol{y}_{i^+}^r$ (respectively $\boldsymbol{y}_{i^-}^r$) the polarity for the positives (respectively negatives) interactions for the reply task:

$$\begin{aligned}
&\mathcal{L}_{class}^{book}(\mathbf{w}_2, \mathcal{S}) \\
&\quad = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{(i^+,i^-) \in \mathcal{I}_u^+ \cup \mathcal{I}_u^-} \Big[\frac{1}{2} \mathbb{1}_{\boldsymbol{y}_{i^+}^b \langle \mathbf{w}_2, (u,i^+) \rangle < 0} + \frac{1}{2} \mathbb{1}_{\boldsymbol{y}_{i^-}^b \langle \mathbf{w}_2, (u,i^-) \rangle < 0} \Big] \\
&\quad \approx \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{(i^+,i^-) \in \mathcal{I}_u^+ \cup \mathcal{I}_u^-} \Big[\frac{1}{2} \log(1 + e^{-\boldsymbol{y}_{i^+}^b \langle \mathbf{w}_2, (u,i^+) \rangle}) \\
&\qquad + \frac{1}{2} \log(1 + e^{-\boldsymbol{y}_{i^-}^b \langle \mathbf{w}_2, (u,i^-) \rangle}) \Big] + \lambda_2 ||\mathbf{w}_2||_2^2 \,,
\end{aligned}$$

$$
\begin{aligned}
&\mathcal{L}^{reply}_{class}(\mathbf{w}_3, \mathcal{S}) \\
&\quad = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{(i^+,i^-) \in \mathcal{I}_u^+ \cup \mathcal{I}_u^-} \left[\frac{1}{2} \mathbb{1}_{\boldsymbol{y}^r_{i^+} \langle \mathbf{w}_2, (u,i^+) \rangle < 0} + \frac{1}{2} \mathbb{1}_{\boldsymbol{y}^r_{i^-} \langle \mathbf{w}_3, (u,i^-) \rangle < 0} \right] \\
&\quad \approx \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{(i^+,i^-) \in \mathcal{I}_u^+ \cup \mathcal{I}_u^-} \left[\frac{1}{2} \log(1 + e^{-\boldsymbol{y}^r_{i^+} \langle \mathbf{w}_3, (u,i^+) \rangle}) \right. \\
&\qquad \left. + \frac{1}{2} \log(1 + e^{-\boldsymbol{y}^r_{i^-} \langle \mathbf{w}_3, (u,i^-) \rangle}) \right] + \lambda_3 ||\mathbf{w}_3||_2^2 \,.
\end{aligned}
$$

Finally, the representation learning task aims at automatically learning an embedding for users $\mathbb{U}$ and an embedding for items $\mathbb{I}$. We insist on two important points here. The first one is that the embedding representations are trained jointly with the other tasks and are used also as inputs in the other tasks. Secondly, in a sense, the embeddings are *specialized* for the click task, as we select pairs with different polarity for the clicks. We consider that the embedding is provides a mistake if

$$
\mathbb{1}_{\langle \mathbb{U}_u, \mathbb{I}_{i^+} \rangle < \langle \mathbb{U}_u, \mathbb{I}_{i^-} \rangle} = \mathbb{1}_{\langle \mathbb{U}_u, \mathbb{I}_{i^+} - \mathbb{I}_{i^-} \rangle < 0} \,,
$$

where $\mathbb{U}_u \in \mathbb{R}^d$ denotes the vector representation for user u and $\mathbb{I}_i \in \mathbb{R}^d$ denotes the vector representation for item i. Again, we use the logistic loss as an approximation to this error and we have

$$
\begin{aligned}
\mathcal{L}^{emb}_{rank}(\mathbb{U}, \mathbb{I}, \mathcal{S}) &= \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{(i^+,i^-) \in \mathcal{I}_u^+ \cup \mathcal{I}_u^-} \mathbb{1}_{\langle \mathbb{U}_u, \mathbb{I}_{i^+} - \mathbb{I}_{i^-} \rangle < 0} \\
&\approx \frac{1}{|\mathcal{U}|)} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{(i^+,i^-) \in \mathcal{I}_u^+ \cup \mathcal{I}_u^-} \log(1 + e^{-\langle \mathbb{U}_u, (\mathbb{I}_{i^+} - \mathbb{I}_{i^-}) \rangle} \\
&\quad + \lambda_4 (||\mathbb{U}_u||_2^2 + ||\mathbb{I}_{i^+}||_2^2 + ||\mathbb{I}_{i^-}||_2^2) \,.
\end{aligned}
$$

4.1 SGD for Multi-Output and Heterogeneous Tasks

As a result of the multiple loss functions defined above, we can write our heterogeneous multi-target learning problem as a constrained convex optimization where the goal is to minimize the general loss functions defined over the sum of all losses defined above and its regularization parameters:

$$
\mathcal{L}(\mathcal{F}, S) + \lambda \Omega(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, u, i^+, i^-) \rightarrow \min \,. \tag{6}
$$

Algorithm 1 describe the SGD strategy we propose to optimize the parameters of our models.

The algorithm works as follow. First, we provide the hyper-parameters values: two stopping criteria parameters, ϵ and the maximum number of epochs

#epochs. We also provide the learning rate η, the number of iteration per epoch *#iters* and the $\lambda_1, \lambda_2, \lambda_3$ and λ_4 parameters that control the regularization terms for respectively the clicks, bookmarks, replies and learning of the embeddings. Then, we randomly initialize all the weights of the models $\mathbf{w}_1, \mathbf{w}_2$ and $\mathbf{w}_3$ and the embeddings for the users $\mathbb{U}$ and for the offers $\mathbb{I}$.

Then, the idea is the following, at each step of the algorithm, the learning is carried out by computing the gradient of each tasks simultaneously and updating the weights of each models, and the weights of the representation $\mathbb{U}$ and $\mathbb{I}$. As the embeddings are initialized randomly, the first iterations of the algorithm mainly relies on the handcrafted features. After few iterations, as quality the embeddings get better, the models rely on both, the handcrafted and the embeddings features. Finally, after updating models and embeddings, we compute the new general loss and compare it to the old one to evaluate if we need to continue the descent.

Algorithm 1 Multi-target learning based on SGD algorithm

Inputs:

$(extracted, context, \mathbb{U}, \mathbb{I}) \in \mathcal{S}$ #Training set
η (Learning rate), $\lambda_1, \ldots, \lambda_4$ (Regularization)
ϵ # Stopping criterion
nb_epochs # Maximum number of epochs
nb_iters # Number of iterations per epochs

Initialize:

Randomly intialized: $W^{(0)} = \{\mathbf{w}_1^{(0)}, \mathbf{w}_2^{(0)}, \mathbf{w}_3^{(0)}, u^{(0)}, i^{+(0)}, i^{-(0)}\}$
$epoch = 0$
$global\_loss\_old \leftarrow 0$
$global\_loss\_new \leftarrow \mathcal{L}(\mathcal{F}, S) + \lambda\Omega(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, u, i^+, i^-)$
while $global\_loss\_new - global\_loss\_old > \epsilon$ and $epoch < nb\_epochs$ **do**
    $global\_loss\_old \leftarrow global\_loss\_new$
    $local\_loss \leftarrow 0$
    for $t = 1...nb\_iters$ **do**
        `randomly choose:` user u, positive offer i^+ and negative offer i^-
                for the click interaction
        $W^{(t)} \leftarrow W^{(t-1)} - \eta \cdot [\nabla\mathcal{L}(\mathcal{F}, S) + \lambda\Omega(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, u, i^+, i^-)]$
        $local\_loss^{(t)} \leftarrow local\_loss^{(t-1)} + [\boldsymbol{\ell}_{rank}^{click}(u, i^+, i^-)] + [\boldsymbol{\ell}_{rank}^{emb}(\mathbb{U}_u, \mathbb{I}_{i^+}, \mathbb{I}_{i^-})]$
            $+[\frac{1}{2}\boldsymbol{\ell}_{class}^{book}(u, i^+) + \frac{1}{2}\boldsymbol{\ell}_{class}^{book}(u, i^-)] + [\frac{1}{2}\boldsymbol{\ell}_{class}^{reply}(u, i^+) + \frac{1}{2}\boldsymbol{\ell}_{class}^{reply}(u, i^-)]$
            $+\lambda_1||\mathbf{w}_1||_2^2 + \lambda_2||\mathbf{w}_2||_2^2 + \lambda_3||\mathbf{w}_3||_2^2 + \lambda_4(||\mathbb{U}_u||_2^2 + ||\mathbb{I}_{i^+}||_2^2 + ||\mathbb{I}_{i^-}||_2^2)$
    end for
    $global\_loss\_new \leftarrow \frac{local\_loss}{nb\_iters}$
    $epoch = epoch + 1$
end while

Remark 2. Here we used l_2 regularization for all loss functions and bind the task using the embedding. The use of l_2 regularization is a decision that can be

debated, however, it is not the focus of this study. Here the goal is to evaluate the impact of learning jointly a representation and different models.

Remark 3. In MTL, when one want to optimize over multiple tasks jointly, the question of the stopping criteria arise. For the SGD algorithm at hand, we defined two different stopping criteria: (i) ϵ is used to measure the losses difference between two epochs. If the difference is lower than a given threshold, say $\epsilon = 10^{-3}$, then we stop the descent and keep the weights. In our MTL scenario, we set ϵ to be a shared criteria accross all tasks. It means that some tasks might continue to update, even if they reached a point where the difference in losses between 2 epochs is lower than ϵ. In other words, the task that takes the slowest task to converge is going to set the number of iteration of the whole optimization. (ii) max_iters is used as a safeguard, in the case where the ϵ does not stop the algorithm and defines the maximum number of iteration.

5 Experiments

In order to evaluate the proposed framework and the quality of the feature extraction method, we conducted a series of experiments aimed at showing the benefit of the jointly learning the representations and the models with SGD strategy. We release the source code[4] and the dataset for research purposes. The aim of the experiments are twofold: (a) to empirically assess if the tasks are effectively interdependent; (b) to evaluate the benefits of learning the tasks joinlty.

Setup and evaluation measures. In both frameworks, MTL and STL, we used ℓ_2 regularized Logistic Regression implemented using SGD algorithm in order to minimize the loss functions for the classification and pairwise ranking tasks. The tuning of hyper-parameters has been made by cross validation over the F1-measure of 3 hyper-parameters: the regularization parameters λ in the range $[10^{-1}, 10^{-4}]$, the class weights in the set $\{1, 3, 5, 7, 9\}$ and the decision threshold probability in $\{0.3, 0.35, 0.4, 0.45, 0.5\}$.

We evaluated the ranking results using two metrics. First, we used the area under the ROC curve (AUC) averaged over all users, and the Mean Average Precision at k (MAP@k). In the case of highly imbalanced datasets, a classical classification metric is the F1-measure that is defined as the harmonic mean of precision and recall.

Implementation details and running time. In our implementation, the feature space is a n-dimensional space, $\mathbf{x} \in \mathbb{R}^n$. n consists in 15 extracted statistical features (see Section 2), 16 contextual features directly extracted from the original dataset, 30 features for the user embedding and 30 features for the offers embedding. Thus, $(w_1, w_2, w_3) \in \mathbb{R}^{91}$, $\mathbb{U} \in \mathbb{R}^{\#\text{users} \times 30}$ and $\mathbb{I} \in \mathbb{R}^{\#\text{offers} \times 30}$. The stopping criteria is set to $\epsilon = 10^{-}3$, the learning rate $\eta = 10^{-6}$ and $\#max_iters = 100$. We set the number of iterations for each epochs to be the product between the

[4] https://github.com/asarbaev/Multi-Target-learning

		Single task				Multitask			
		F_1	AUC	MAP@1	MAP@5	F_1	AUC	MAP@1	MAP@5
Ranking	Clicking	$0.51^{\downarrow}$	$0.62^{\downarrow}$	$0.659^{\downarrow}$	$0.537^{\downarrow}$	**0.54**	**0.65**	**0.708**	**0.584**
	Embedding	0.42	$0.52^{\downarrow}$	**0.427**	**0.318**	**0.46**	**0.54**	0.421	0.314
Classification	Bookmarked	$0.03^{\downarrow}$	$0.51^{\downarrow}$	$0.008^{\downarrow}$	$0.005^{\downarrow}$	**0.06**	**0.54**	**0.026**	**0.015**
	Replied	**0.17**	$0.55^{\downarrow}$	$0.062^{\downarrow}$	$0.038^{\downarrow}$	0.16	**0.58**	**0.083**	**0.05**

Fig. 2: Results on classification (F1-measure) and pairwise ranking (AUC) averaged over users for the single-task and the multi-task/stacking strategies on the `DAEMON` dataset. The best results are shown in bold, and a $\downarrow$ indicates a result that is statistically significantly worse than the best, according to a Wilcoxon rank sum test with $p < 0.01$.

number of unique users, the minimum number of positive interacted offers and the minimum number of negative interacted offers as

$$\#iters = \#unique\_users \times \#min\_nb\_pos \times \#min\_nb\_neg,$$

where $\#unique\_users$ is the number of unique users in dataset, $\#min\_nb\_pos$ (respectively $\#min\_nb\_neg$) represents lowest number of positives (resp. negatives) clicks interactions that a user can have.

We implemented the SGD described in Algorithm 1 in Python for both, the single task learning and MTL models. The computations for 1 epoch takes about 10 minutes on a single 3.2 GHz core, that is about 6.5 hours for the training of all models for MTL when considering a learning rate of $\eta = 10^{-6}$ and a stopping criteria set to $\epsilon = 10^{-3}$.

Models performance analysis. Table 2 presents the results for all the tasks and both approaches, heterogeneous multi-target learning and single task learning. We use a bold font to highlight the highest performance rates and a $\downarrow$ to show that a performance is significantly worse than the best result, according to a Wilcoxon rank sum test used at a p-value threshold of 0.01 [11]. First, and without any surprise, the predictions performance are better for the balanced tasks, in terms of positives and negatives examples. In both setups, the predictions of clicks provides better results than any other tasks, up to 0.65 in terms of AUC and 0.708 in terms of $MAP@1$. In most cases, we observe that multi-task ranking approach achieves statistically significant improvements compared to the single-task learning approach. While quite close regarding $MAP@k$ measure, multi-task improves AUC measure consistently for all the tasks, apart from the embedding learning tasks. Intuitively, the learning of the embedding in the single task learning case is specialized for the clicks task while in the multitask case it bias all tasks. Regarding F1-measure, the multi-target learning approach is also better in all tasks but in the case of replies.

6 Conclusions & Future Work

In this paper, we consider the problem of MTL dyadic prediction in a recommendation systems setting. Based on the collection of the RecSys 2016 challenge for

a job recommendation, we propose a method to extract meaningful dyads representation based on multiple implicit feedbacks and extracted `DAEMON` a dataset for this task. We also propose and implement a SGD algorithm that learns jointly over multiple heterogeneous tasks. To the best of our knowledge, this work is the first to learn an embedding jointly with multiple heterogeneous tasks using a SGD approach. We show that this algorithm allows a substantial improvement over the single-target learning case. These results also bring evidence that the proposed dataset is of interest for the problem at hand. An interesting starting point for future work would be to extend the proposed heterogeneous SGD algorithm with a shared regularization.

References

1. Bennett, J., Lanning, S.: The Netflix Prize. In: KDD Cup and Workshop 2007, p. 35 (2007)
2. Ben-David, S., Schuller, R.: Exploiting Task Relatedness for Multiple Task Learning. In: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, pp. 567–580. Springer, Berlin (2003)
3. Caruana, R.: Multitask Learning. Machine Learning. 28 (1), 41–75 (1997)
4. Yang, X., Seyoung, K., Xing, E. P.: Heterogeneous Multi-Task Learning with Joint Sparsity Constraints. In: Advances in Neural Information Processing Systems 22, pp. 2151–2159, Vancouver (2009)
5. Sculley, D.: Combined Regression and Ranking. In: 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 979–988, Washington (2010)
6. Kumar, A., Daume, H.: Learning Task Grouping and Overlap in Multi-Task Learning. In: 29th International Conference on Machine Learning, pp. 1383–1390. New York (2012)
7. Chapelle, O., Shivaswamy, P., Vadrevu, S., Weinberger, K., Zhang, Y., Tseng, B.: Multi-Task Learning for Boosting with Application to Web Search Ranking. In: 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1189–1198, Washington (2010)
8. Evgeniou, T., Pontil, M.: Regularized Multi-Task Learning. In: 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 109–117, New York (2004)
9. Stock, M., Pahikkala, T., Airola, A., De Baets, B., Waegeman, W.: Efficient Pairwise Learning using Kernel Ridge Regression: an Exact Two-Step Method. Technical report (2016)
10. Volkovs, M., Zemel, R.S.: Collaborative Ranking with 17 Parameters. In: Advances in Neural Information Processing Systems 25, pp. 2294–2302, Lake Tahoe (2012)
11. Lehmann, E. L., Romano, J.P.: Testing Statistical Hypotheses. Springer Texts in Statistics, New York (2005)