Grenoble Institute of Technology

Master of Science in Industrial and Applied Mathematics

Data Science

Multi-target learning for Implicit Feedback in Recommendation Systems

Author:
Amir Asarbaev

Supervisor:
Massih-RezaAmini
Yury Maximov
Simon Moura

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'avenir.

Contents

Abstract

In this research work 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 multiple implicit feedback for RS.

1 Introduction

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

In this work, we consider the case where there are multiple implicit feedbacks on items and propose a simple yet effective learning algorithm to enhance prediction over each of the tasks. 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 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, 5, 6, 7]. Most of these approaches learn a different model for each task

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

using the feature representation of observation and model the dependencies in the objective function with 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 research work, 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 a 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 feedback. 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 multitask gradient descent 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 multitask 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 2.1 we briefly present the RecSys 2016 challenge dataset and the features that were extracted from the implicit feedback. In Section 2 we describe the multitask learning framework considered throughout this research work and `Stochastic Gradient Descent` for jointly learning multiple heterogeneous tasks and embeddings.In Section 4 we present experimental results that evaluate the effect of taking into account the dependency among the outputs. Finally, in Section 5, we discuss the outcome of this study and give some pointers for further research.

2 Multi-task Framework

This section provides a formal definition of the multi-target learning framework. First, we define the notations and the classical single target learning framework. Then we describe the multi-target framework and its originality regarding close setups such as multi-task learning originally proposed by Caruana in 1997 [3].

In the remaining part of this article, $\mathcal{Y}$ refers to the output of the learning model and $\mathcal{X}$ designates the set of training examples available for learning. A training set $\mathcal{S} = (x_i, y_i)_{i=1}^m$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, is composed of examples randomly sampled following a fixed but unknown probability distribution $\mathcal{D}$. $\mathcal{H}$ is the considered function space and $h \in \mathcal{H}$ is a specific hypothesis in this space function.

In single task learning (STL), the goal is to learn an hypothesis function $h : \mathcal{X} \to \mathcal{Y}$ that maps examples to a *simple* output $y_i \in \mathcal{Y}$. The main goal is to learn an hypothesis function h such that it approximates $\mathcal{D}$ in the best possible way.

Typical STL frameworks are multi-class classification where the output space is $\mathcal{Y} = \{1, \ldots, K\}$ and where $1, \ldots, K$ represents the K possible labels for an input $\mathbf{x} \in \mathcal{X}$. In the case of binary classification, the output space is typically $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, +1\}$. In regression tasks, one try to map an example $\mathbf{x} \in \mathcal{X}$ to a real value, $\mathcal{Y} = \mathcal{R}$.

The classical approach to solve single task learning problems is consider a function $h : \mathcal{X} \to \mathcal{Y}$ that map examples of the training set $\mathcal{X}$ to one of the two possible output $y \in \mathcal{Y}$. To do so, one use algorithms that *learn* the function h by solving an optimization problem based on the error made by the predictor.

In order to fit the parameters of the model to a training set $\mathcal{X}$, a common approach is to minimize a *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. This function measures the quality of a prediction by attributing a *cost* that compares the true label to the predicted one.

In multi-target learning (MTL) the aim is to extend the STL framework by generalizing its framework to handle multiple related heterogeneous tasks.

Formally, MTL aims at solving learning problems where the output space can be written as $\mathcal{Y} = \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_T$ and where $\mathcal{Y}_1 \times \ldots \times \mathcal{Y}_T$ can be tasks of different type such as classification, regression, ranking and/or any other output of a learning problem. Contrary to multi-label learning or multi-output regression, the number of output is fixed in advance and the output can be heterogeneous. It is important to note that if the number of task is 1, we are in the case of STL. Thus, all properties proven in the general case of MTL will still apply in the case of STL.

The goal here is to find a mapping h between a given input space $\mathcal{X}$ to multiple output space of possibly different type: $h : \mathcal{X} \to \mathcal{Y} = \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_T$. In order to solve a multi-output interdependent heterogeneous problem, multiple approaches can be considered:

1. **Dataset modification**: split the dataset into single tasks problems and solve them separately. This approach is used to solve multiple complex problems such as multi-label learning [11]. A major drawback is that this method does not take into account the dependency between tasks;

2. Find a **shared decision function** h which take a decision for all tasks considered simultaneously. For instance, if we consider two tasks: t_1 a binary classification task and t_2 a regression task, one aim at finding a mapping $h \rightarrow \mathcal{Y}_1 \times \mathcal{Y}_2$, where $\mathcal{Y}_1 = \{0, 1\}$ and $\mathcal{Y}_2 = \mathbb{R}$;

3. Find **multiple decision function** $h = (h_1, h_2, \ldots, h_T)$ which take advantages of the interdependency and the model learnt on other tasks;

2.1 The recommendation application

In the following 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].

2.2 RecSys 2016 datasets

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:

- `users.csv` consists of contextual information about each user.
- `items.csv` consists of contextual information about each item.
- `impressions.csv` consists of information about what offers were displayed for each user.
- `interactions.csv` consists of information what interactions were done by each user with respect to displayed offers for them. Within the framework of this challenge only 4 types of interactions were provided: *clicked*, *bookmarked*, *replied* and *deleted*.

In this collection, users provided implicit feedback and may had different interactions with a same offer: for instance 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.

[2]We make available the extracted dataset as well as the codes for research purpose.

[3]`https://recsys.acm.org/recsys16/challenge/`

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 our primary goal. The statistics of the original interactions and impressions are summarized in Table 1 (left). In this table, sparsity represents the user-offer pairs for which there is no interaction at all.

After exploring `interactions.csv` and `impressions.csv` files there was realized that we had a highly imbalanced case. Fig.1 shows the number of offers displayed to users, as well as the number of offers clicked, bookmarked and replied. It comes out that the offers which are bookmarked and replied, represent less than 5% of those that are displayed to the users, making both associated tasks challenging for classification. From this observation, we defined a Multi-Target scenario for the following tasks:

- pairwise approach of learn-to-rank problem for clicking interactions, where we target to provide a ranking list of offers on which a user is willing to click.
- two binary classification problems, where the goal is to predict if a user is willing to interact positively with specific offers regarding bookmarks and reply.

Accordingly, for the solution of defined tasks, there was raised a problem of the representation of pairs. The next subsection will be devoted to our solution of this problem.

2.3 Feature extraction based on implicit feedback

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

2.3.1 Data preparation

Due to the large amount of data available and the substantial sparsity, we subsampled the dataset with respect to users activity (number of unique offers were interacted by target user) and offers popularity(number of unique users interacted with target offer) in order to only keep users for which we had enough information to extract meaninful statistics. We decided to keep users with more than 30 interactions and offers which had been interacted with at least 30 times. Following

	RecSys'16	DAEMON
# Users	**770.859**	**5.949**
# Offers	**1.002.161**	**25.184**
# Observations	**10.378.780**	**819.226**
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).

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 by taking into account time line of interactions (i.e. the test set consists mostly latest (with respect to time) interactions). Table 1 (right) 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 which did not interact enough are not relevant to learn a predictive model as we lack information about them. Secondly the method we use to extract features [10] heavily relies on the quality of the similarity between users which is lowered by a highly sparse dataset.

2.3.2 Neighbours preferences

In order to compute the statistics of [10], there was involved information from `impressions.csv` and `interactions.csv` files. We used two different representations of user-offer interactions:

- In one case, we binary classify dyadic(user-offer) pairs for distinguishes between pairs in which offers displayed without interactions and those for which there were at least one interaction (clicked, bookmarked or replied).

- In another case, we associated to clicked, bookmarked and replied respec-

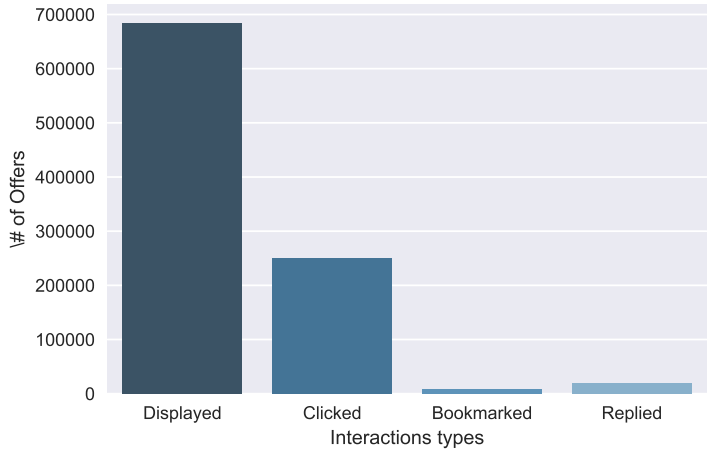

Figure 1: Number of offers, *displayed*, *clicked*, *bookmarked* and *replied*. *All* is the sum of the three latter. The dataset is highly imbalanced, it contains only few *Bookmarked* and *Replied* interactions.

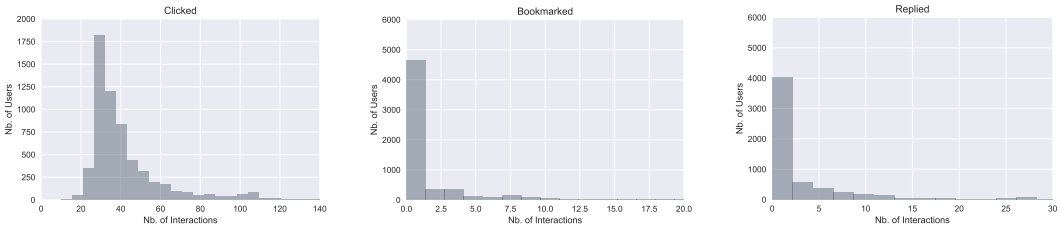

Figure 2: The number of users with respect to the number of interactions. From left to right, for *Clicked*, *Bookmarked* and *Replied* tasks.

tively the integers 1, 2 and 3 that could describe the extent of the user's interest to offers.

Fig.2 shows, the number of users with respect to the number of interactions for the three tasks clicked, bookmarked and replied. As expected, the vast majority of users make very few interactions while few users interact a lot; and the phenomena is accentuated for the two last tasks (bookmarked and replied). It was a reason to use binary representation approach on the stage of neighbors selection. There was applied the nearest neighbors algorithm with cosine distance to define a set of most similar users to target user by taking into account binary similarity with respect to interacted offers.

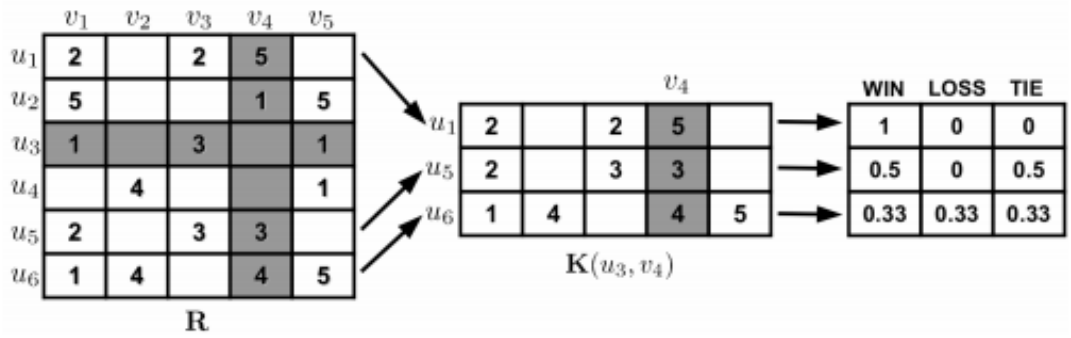

Figure 3: An example rating matrix R and the resulting WIN, LOSS and TIE vectors for the user-item pair (u_3, v_4) with K = 3 (number of neighbors).(1) Top-3 closest neighbors $\{u_1, u_5, u_6\}$ are selected from $U(v_4) = \{u_1, u_2, u_5, u_6\}$ (all users who rated $v_{\{}4\}$). Note that u_2 is not selected because the ratings for u_2 deviate significantly from those for u_3. (2) The WIN, LOSS and TIE matrices are computed for each neighbor using Equation 1. Here g $\equiv 1$ (i.e. there is ignore rating magnitude) is used to compute the matrices. For example, u_5 gave a rating of 3 to v_4 which ties it with v_3 and beats v_1. Normalizing by $|V(u_5)| - 1 = 2$ gives $WIN_{34}(u_5) = 0.5$, $LOSS_{34}(u_5) = 0$ and $TIE_{34}(u_5) = 0.5$.

2.3.3 Extraction of the statistical features

Following to steps described in the NIPS algorithm [10], for each dyad (u,o) we use all of the observed ratings for each neighbor and summarize the net preference for target offer into three preference vectors `win/tie/loss`. It is important to note that at this stage we use associated representation approach in which each type of interaction corresponds to a natural integer. Hence, the mentioned above rating is the sum of these integers. For instance we have dyad (u_n, o_m) in which user $\mathtt{u_n}$ clicked on the offer $\mathtt{o_m}$ and after he saved this offer in the bookmarks, it gives us {1,2,0} and rating will be equal $r = 1 + 2 + 0 = 3$. Figure 3 describe the concept of computational `win/tie/loss` vectors.

$$WIN_{nm}(k) = \frac{1}{|V(u_k) - 1|} \sum_{V(u_k) \setminus v_m} g(\mathbf{R}(u_k, v_m, \mathbf{R}(u_k, v'))I[\mathbf{R}(u_k, v_m) > \mathbf{R}(u_k, v')]$$

$$LOSS_{nm}(k) = \frac{1}{|V(u_k) - 1|} \sum_{V(u_k) \setminus v_m} g(\mathbf{R}(u_k, v_m, \mathbf{R}(u_k, v'))I[\mathbf{R}(u_k, v_m) < \mathbf{R}(u_k, v')]$$

$$TIE_{nm}(k) = \frac{1}{|V(u_k) - 1|} \sum_{V(u_k) \setminus v_m} I[\mathbf{R}(u_k, v_m) = \mathbf{R}(u_k, v')] \quad (1)$$

Using computed `win/tie/loss` vectors we extracted 15 simple statistical fea-

tures (five for each vector) Equation (2).

$$\gamma(WIN_{nm}) = \Big[\mu(WIN_{nm}), \sigma(WIN_{nm}), max(WIN_{nm}), min(WIN_{nm}), \frac{1}{K}\sum_{k} I[WIN_{nm}(k) \neq 0]\Big] \tag{2}$$

where μ is mean, σ is standard deviation,max and min is maximum/minimum value in the vector and last statistical feature is the number of neighbors (out of K) that express any positive preference towards v_m

2.4 Final dataset

Final dataset was built with involving 15 statistical extracted features and 16 contextual features taken from `users.csv` and `items.csv` files. Contextual features include following information:

- for users:
  1. `career level` career level ID (e.g. beginner, experienced, manager)
  2. `discipline id` anonymized IDs represent disciplines such as "Consulting", "HR", etc.
  3. `industry id` anonymized IDs represent industries such as "Internet", "Automotive", "Finance", etc.
  4. `country` describes the country in which the user is currently working
  5. `region` is specified for some users who have as `country` de
  6. `experience n entries class` identifies the number of CV entries that the user has listed as work experiences
  7. `experience years experience` is the estimated number of years of work experience that the user has
  8. `experience years in current` is the estimated number of years that the user is already working in her current job.
  9. `edu degree` estimated university degree of the user
- for offers:
  1. `career level` career level ID (e.g. beginner, experienced, manager)
  2. `discipline id` anonymized IDs represent disciplines such as "Consulting", "HR", etc.
  3. `industry id` anonymized IDs represent industries such as "Internet", "Automotive", "Finance", etc.

4. `country` code of the country in which the job is offered.
5. `region` is specified for some users who have as `country` de
6. `employment` the type of employment (full-time, part-time , etc.)
7. `active during test` is 1 if the item is still active (= recommendable) during the test period and 0 if the item is not active anymore in the test period (= not recommendable)

2.5 The representations

This chapter summarizes all the representations used in the preparation of the database in order to avoid confusion of the reader.

The given dataset consists of contextual features which were mentioned above. The contextual representation of the pairs provided as very poor results, thus we had to find some other way to represent our pairs. We used algorithm described in 2.3. subsection to extract statistical representation of our pairs based on users preferences. At the stages of finding neighbors and comparing their ratings of this algorithm there were used different representation of interactions which are described at the beginning of 2.3.2 section. Thus, there were involved two types of (user-item) representation: statistical and contextual. The combination of these two types of representation showed good results which was the reason for our interest in learning the better representation.

3 Learning objective

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 of dyads and their associated multiple 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

$$\mathcal{L}(\boldsymbol{h}) = \mathbb{E}_{(\mathbf{x},\boldsymbol{y})\sim\boldsymbol{\mathcal{D}}}[\boldsymbol{\ell}(h(\mathbf{x}), \boldsymbol{y})], \tag{3}$$

where $\boldsymbol{\ell}(h(\mathbf{x}), \boldsymbol{y})$ is an instantaneous loss measuring the discrepancy of the predictions over different outputs $\boldsymbol{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_T(\mathbf{x})) \in \mathcal{Y}$ of observation $\mathbf{x}$ and its desired output $\boldsymbol{y} = (y_1, \ldots, y_T) \in \mathcal{Y}$. Following the Empirical Risk Minimization principle, we will achieve this aim by minimizing an empirical loss over a training set of size $m : \mathcal{S} = (\mathbf{x}_i, \boldsymbol{y}_i)_{i=1}^m$, where examples are also supposed to be generated i.i.d with respect to the same probability distribution $\boldsymbol{\mathcal{D}}$.

$$\mathcal{L}_m(h, \mathcal{S}) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{T}\sum_{j=1}^{T}\boldsymbol{\ell}_j(h_j(\mathbf{x}_i), \boldsymbol{y}_i) + \lambda\Omega(\boldsymbol{h}), \tag{4}$$

where $\Omega(\boldsymbol{h})$ is a regularization term that joints all the predictions.

In this work 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.

Each task is defined by the average of all logarithmic losses in the samples. In the case of pairwise ranking we note by $\mathcal{U} \subseteq N$ (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 $(i, u, i') \in \mathcal{I}_u^- \times U \times \mathcal{I}_u^-$ where:

$$y_{i,u,i'} = \begin{cases} 1 & if\ i \underset{u}{\succ} i' \\ -1 & otherwise \end{cases} \tag{5}$$

Hence, we can write the ranking objective function for the clicks as follow:

$$\begin{aligned} \mathcal{L}_R^c(w_1, S) &= \frac{1}{|U|} \sum_{u \in U} \frac{1}{|J_u^+||J_u^-|} \sum_{i^+ \in J_u^+} \sum_{i^- \in J_u^-} \mathbb{1}_{f_1(u,i^+) < f_1(u,i^-)} \\ &\approx \frac{1}{|U|} \sum_{u \in V} \frac{1}{|J_u^+||J_u^-|} \sum_{(i,i') \in J_u^+ \cup J_u^-} log(1 + e^{-y_{i,u,i'} < w,(u,i) - (u,i') >}) + \lambda ||w_1||_2^2 \end{aligned}$$

Remark. *Note that, for a fixed user, it is possible to have a different polarity of interactions with respect to clicks, bookmarks and reply. 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 bookmark or reply.*

In the case of the classification tasks and considering Remark 3, we compute the logistic loss using the polarity of the labels corresponding to either, the bookmark or the reply task. Also note that in the case of classification at each iteration, we consider the average of the losses for both items.

$$\begin{aligned} \mathcal{L}_C^b(w_2, S) &= \frac{1}{|U|} \sum_{u \in U} \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} \mathbb{1}_{y_{u,i}^b f_2(u,i) < 0} \\ &\approx \frac{1}{|U|} \sum_{u \in U} \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} \frac{1}{2} \log(1 + e^{-y_{u,i}^b \langle w_2, (u,i) \rangle}) + \lambda ||w_2||_2^2 \end{aligned}$$

$$\begin{aligned} \mathcal{L}_C^r(w_3, S) &= \frac{1}{|U|} \sum_{u \in U} \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} \mathbb{1}_{y_{u,i}^r f_3(u,i) < 0} \\ &\approx \frac{1}{|U|} \sum_{u \in U} \frac{1}{|\mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} \frac{1}{2} \log(1 + e^{-y_{u,i}^r \langle w_3, (u,i) \rangle}) + \lambda ||w_3||_2^2 \end{aligned}$$

Finally, the representation learning task aims at learn representations $u \in \mathcal{U}$ and items $i, i' \in \mathcal{I}$ such that in the embedded space of dimension d the preference is preserved through the dot product.

$$\mathcal{L}_R^e(\mathbb{U}, \mathbb{I}, S) = \frac{1}{|U|} \sum_{u \in U} \frac{1}{|J_u^+||J_u^-|} \sum_{i^+ \in J_u^+} \sum_{i^- \in J_u^-} \mathbb{1}_{y_{i,u,i'}(\mathbb{UI} - \mathbb{UI}') < 0}$$

$$\approx \frac{1}{|U|} \sum_{u \in U} \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{(i,i') \in \mathcal{I}_u^+ \cup \mathcal{I}_u^-} \log(1 + e^{-y_{i,u,i'} < u, (i - i') >}) + \lambda(||u||_2^2 + ||i||_2^2 + ||i'||_2^2)$$

Following the problems description above, we can re-write the general loss function in Eq. (4) as follow:

$$\widehat{\mathcal{L}}_m(h, S) = \mathcal{L}_R^c(f_1, S) + \widehat{\mathcal{L}}_C^b(f_2, S) + \widehat{\mathcal{L}}_C^r(f_3, S) + \mathcal{L}_R^e(f_4, \mathbb{U}, \mathbb{I}) + \lambda\Omega(\phi), \tag{6}$$

where $h = (f_1, f_2, f_3, f_4)$, $\phi = w_1, w_2, w_3, u, i^+, i^-$ and $\lambda > 0$ is a regularization parameter.

To solve optimization task (6), we choose Stochastic Gradient Descent algorithm detailed in Algorithm 3.2, because it results in low computational complexity.

3.1 The convergence proof of the SGD

The objective function given by Eq. (3) is non-convex and challenging in to optimize in high-dimensions. The Randomized Stochastic Gradient Descent is the most common tool to solve a large-scale stochastic optimization problem to the local optimality and even escape from small local minimums [12]. Below we assume that a set of functions $\mathcal{H}$ is parametrized by parameter $\omega \in W$ so that optimal $\boldsymbol{h} \in \mathcal{H}$ minimizes the expected risk, Eq. (3), corresponds to $\boldsymbol{h}_{\omega *}$. As it is often the case, that set W is non-convex we provide an analysis of the stochastic gradient convergence to minimize the expected risk of multi-target learning for the non-convex case.

Recall, that the Randomized Stochastic Gradient (RSG) descent [12] in our setting have the form

1. For each $k = 1, ..., N$ update the current value of ω with a stochastic subgradient G_k of $\mathbb{E}_{(\mathbf{x},\boldsymbol{y})\sim\boldsymbol{\mathcal{D}}}\mathcal{L}(\omega) + \lambda\Omega(\omega)$;
2. Update $\omega^{k+1} = \omega^k - \gamma_k G_k(\omega^k)$, where $\gamma_k > 0$ is a step-size.
3. Return ω^k, $1 \leq k \leq N$, with probability $1/N$.

We have

Theorem 1. *Suppose that the stepsizes* $\{\gamma_k\}_{k=1}^N$*, and sub-gradients* $\{G_k\}_{k=1}^N$ *are chosen such that for any* k*,* $1 \leq k \leq N$*,* $\mathbb{E}_{(\mathbf{x},\boldsymbol{y})\sim\boldsymbol{\mathcal{D}}} G_k(\boldsymbol{h}(\phi^k)) = \nabla\mathcal{L}(\boldsymbol{h}(\phi^k))$*, and*

$$\gamma_k = \min\left\{\frac{1}{L}, \frac{D}{\sigma\sqrt{N}}\right\} \qquad \mathbb{E}_{(\mathbf{x},\boldsymbol{y})\sim\boldsymbol{\mathcal{D}}}\|G_k(\boldsymbol{h}(\omega^k)) - \nabla f(\boldsymbol{h}(\omega^k))\|_2^2 \leq \sigma^2.$$

Then

$$\mathbb{E}_{(\mathbf{x},\boldsymbol{y})\sim\boldsymbol{\mathcal{D}}}\|\nabla\mathcal{L}(\boldsymbol{h}(\omega_r))\|_2^2 \leq \frac{LD^2}{N} + \frac{2D\sigma}{\sqrt{N}},$$

where $D = (2(\mathcal{L}(\boldsymbol{h}(\phi^0)) - \mathcal{L}(\boldsymbol{h}(\phi^*))/L)^{1/2}$*, and* $\mathcal{L}$ *is an* L *smooth function.*

Theorem 1 is an immediate consequence of Theorem 2.1, proved in [12], and states the rate of convergence of the RSG algorithm to a local optimum.

It remains to derive the parameters involved in the statement of Theorem 1. For the optimization problem, the smoothness constant, L, is bounded from the above by the sum of smoothness constants of the classification and ranking losses, and the regularization term. Also, a variance, σ^2, is bounded from the about by $4L^2$ by the additivity of the loss function, where L is a smoothness constant of $\mathcal{L}$. Thus, all the parameters used in Theorem 1 are bounded regarding the function smoothness and desired accuracy allows establishing an efficient bound on the number of steps required by the randomized stochastic gradient descent to converge to a local optimum.

3.2 The algorithm

Algorithm 1 Multi-target learning based on SGD algorithm

Inputs:
training set: $(extracted, context, \mathbb{U}, \mathbb{I}) \in S$
learning rate η; regularization parameter λ;
number of iterations N

Initialize:
random weights vector $\phi^0 = \{w_1^0, w_2^0, w_3^0, u^0, i^{+,0}, i^{-,0}\}$:
$w_1^0, w_2^0, w_3^0 \in \mathbb{R}^{(extr+cont+U+V)}$
$\mathbb{U}^{(0)} \in \mathbb{R}^{(U)}$, $\mathbb{I}^{(0)} \in \mathbb{R}^{(V)}$ is a matrix of users (resp. items) representation

$\mathcal{L}^0 = \mathcal{L}(\phi^0, S) + \lambda\Omega(\phi^0)$

for $t = 1...N$ **do**

$\mathcal{L}_u \leftarrow 0$ // Choose a user uniformly at random, assign associated loss to 0

Randomly choose a pair of positive offer i^+, negative offer i^-, and update:

$$\phi^t \leftarrow \phi^{t-1} - \eta \cdot [\nabla\mathcal{L}(\phi, S) + \lambda\Omega(\phi)]$$

Update the loss function value, where ℓ is loss of each task on each iteration.

$$\mathcal{L}_u \leftarrow \mathcal{L}_u + \ell_R^c(u, i^+, i^-) + \frac{1}{2} \cdot \ell_C^b(u, i^+) + \frac{1}{2} \cdot \ell_C^b(u, i^-) + \frac{1}{2} \cdot \ell_C^r(u, i^+) + \frac{1}{2} \cdot \ell_C^r(u, i^-) + \ell_R^e(u, i^+, i^-) + \lambda\Omega(\phi^t)$$

$\mathcal{L}^t \leftarrow \mathcal{L}^{t-1} + \mathcal{L}_u$

end for

4 Experimental results

To evaluate the proposed framework and the quality of the feature extraction method, we conducted a series of experiments on the `DAEMON` sub-sample dataset for multitask learning. We implemented the algorithm in Python and release the source code for research purposes. Table 1 shows the number of offers displayed to users, as well as the number of offers clicked, bookmarked and replied. It comes out that the offers which are bookmarked and replied, represent less than 5% of those that are displayed to the users, making both associated tasks challenging for classification.

The aim here was twofold: (a) to note if the recommendation tasks are effectively interdependent and, without any bias over the strength of the model in use, is it possible to learn more efficiently by considering them jointly; (b) to see if with classification one can learn a better ranking function for the recommendation tasks considered? To measure the effect of this combination, we also bind the outputs of the classifiers at the first level, by learning a second classification model at the second level of the stacking strategy.

Evaluation measures We evaluated the ranking results using two metrics. First, we used the area under the ROC curve (AUC) measure averaged over all users, defined as :

$$AUC = \frac{1}{|\mathbb{U}|} \sum_{u \in \mathbb{U}} AUC_u$$
$$AUC_u = 1 - (\text{proportion misordered pairs for user u})$$

Where $\mathbb{U}$ represents the set of users.

Secondly, we used average precision at k ($AP_u@k$), that evaluate the average number of correctly ordered offer, up to offer ranked k for a user u:

$$MAP@k = \frac{1}{|\mathbb{U}|} \sum_{u \in \mathbb{U}} AP@k$$
$$AP_u@k = \frac{\#\text{negative offer ranked higher than positive offers}}{k}$$

As the tasks are highly unbalanced, for classification we considered the F1-measure, defined as the harmonic mean of precision and recall:

$$F1 = \frac{2 * (\text{precision} * \text{recall})}{(\text{precision} + \text{recall})}$$

		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	$0.42^{\downarrow}$	$0.52^{\downarrow}$	**0.427**	**0.318**	**0.46**	**0.54**	$0.421^{\downarrow}$	$0.314^{\downarrow}$
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**	$0.55^{\downarrow}$	$0.062^{\downarrow}$	$0.038^{\downarrow}$	$0.16^{\downarrow}$	**0.58**	**0.083**	**0.05**

Figure 4: 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 $p < 0.01$.

Experimental setup In both frameworks, MTL and STL, we used ℓ_2 regularized Logistic Regression implemented using Stochastic Gradient Descent 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 parameter 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 $\{0.3, 0.35, 0.4, 0.45, 0.5\}$.

Notes on implementation of the algorithm 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 2.2), 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 $\mathbb{I} \in \mathbb{R}^{\#\text{offers} \times 30}$. We also fixed the learning rate to $\eta = 10^{-6}$ and the stopping criterion at $\epsilon = 10^{-3}$. If the stopping criterion was not met, we considered the maximum number of iterations (#iters) 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.

Results Fig.5 and Fig.6 shows dependence of the loss function on the iteration number for STL and MTL, respectively. It is important to notice that we used stopping criterion which was equal $\epsilon = 10^{-3}$. As we can see in both cases we have dramatically fast convergence for ranking and classification tasks and slow decrease for embedding loss.

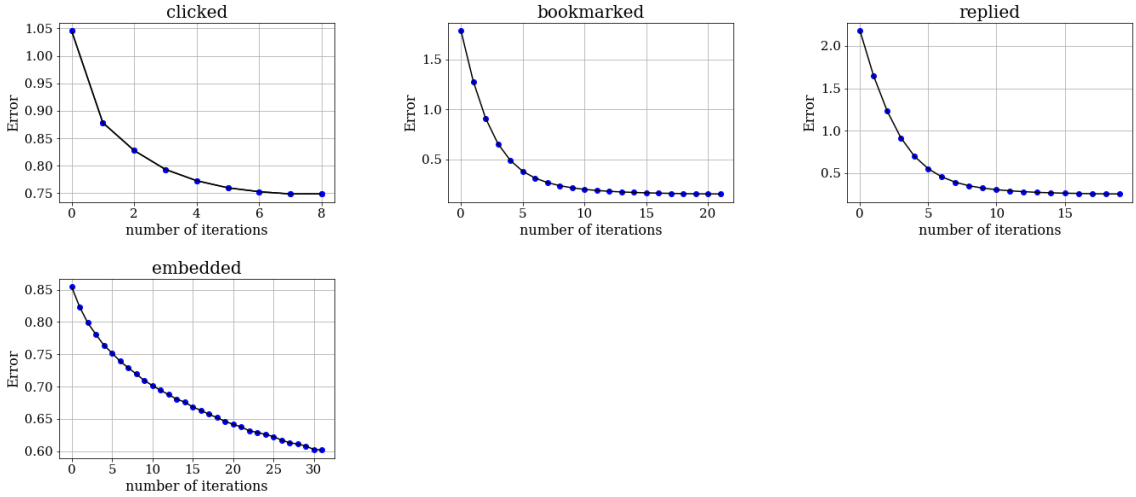

Figure 5: Single-task learning. Proof of convergence. From left to right, for *pairwise ranking* for clicking interactions, *classification* task for bookmarks,*classification* task for replies and *embedding*

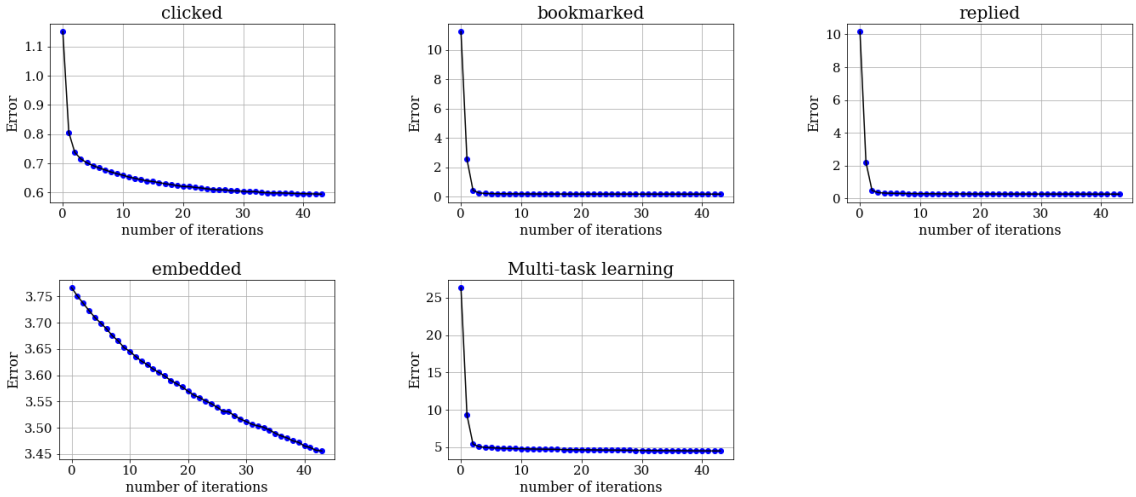

Figure 6: Multi-target learning. Proof of convergence. From left to right, for *pairwise ranking* for clicking interactions, *classification* task for bookmarks, *classification* task for replies, *embedding* and **Multi-target learning**.

Table 4 presents the results for all the tasks and both approaches. We use a bold font to indicate the highest performance rates and a ↓ 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 [13].

Results analysis Without any surprise, the predictions are better on the balanced tasks. In both frameworks, the predictions of clicks provides better results than any other tasks, up to 0.65 in terms 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 only for the clicks 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.

5 Conclusion

In this paper, we consider the problem of multi-task 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 stochastic gradient descent 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 stochastic gradient descent approach. We show that this algorithm allows a substantial improvement over the single-target learning case when the tasks are learned independently. These results also bring evidence that the proposed dataset `DAEMON` is of interest for the problem at hand. An interesting starting point for future work would be to extend the proposed algorithm with a shared regularization.

References

[1] James Bennett and Stan Lanning. The netflix prize. 2007.

[2] Shai Ben-David and Reba Schuller. *Exploiting Task Relatedness for Multiple Task Learning*, pages 567–580. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[3] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[4] Xiaolin Yang, Seyoung Kim, and Eric P. Xing. Heterogeneous multitask learning with joint sparsity constraints. In *Advances in Neural Information Processing Systems 22 (NIPS'09)*, pages 2151–2159. 2009.

[5] D. Sculley. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 979–988. 2010.

[6] Abhishek Kumar and Hal Daume. Learning task grouping and overlap in multi-task learning. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1383–1390, New York, NY, USA, 2012. ACM.

[7] Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Multi-task learning for boosting with application to web search ranking. In *Proceedings of the 16th ACM SIGKDD*, pages 1189–1198, 2010.

[8] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 109–117, New York, NY, USA, 2004. ACM.

[9] Michiel Stock, Tapio Pahikkala, Antti Airola, Bernard De Baets, and Willem Waegeman. Efficient pairwise learning using kernel ridge regression: an exact two-step method. *CoRR*, abs/1606.04275, 2016.

[10] Maksims Volkovs and Richard S. Zemel. Collaborative ranking with 17 parameters. In *Advances in Neural Information Processing Systems 25 (NIPS'12)*, pages 2294–2302. Curran Associates, Inc., 2012.

[11] M. L. Zhang and Z. H. Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, Aug 2014.

[12] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[13] E. L. Lehmann and Joseph P. Romano. *Testing statistical hypotheses*. Springer Texts in Statistics. Springer, New York, third edition, 2005.