Prog Artif Intell (2018) 7:15–29
https://doi.org/10.1007/s13748-017-0133-5

REGULAR PAPER

An adapted incremental graded multi-label classification model for recommendation systems

Khalil Laghmari[1,2] · **Christophe Marsala**[2] · **Mohammed Ramdani**[1]

Received: 9 June 2017 / Accepted: 11 August 2017 / Published online: 24 August 2017

Abstract Graded multi-label classification (GMLC) is the task of assigning to each data a set of relevant labels with corresponding membership grades. This paper is interested in GMLC for large and evolving datasets where data are collected from a possibly infinite stream. Many commercial and non-commercial websites acquire such data by giving users the opportunity to rank items any time using an ordinal scale like one-to-five star rating. Typically these collected data are sparse because users rank only a small subset of items. Websites rely on recommender systems to dynamically adapt the recommended item set for each user. Hence, the applied recommender system should remain scalable and efficient when dealing with sparse data. State-of-the-art methods related to GMLC were tested only in batch mode. Their performance in an incremental mode is not investigated, especially in presence of sparse data and concept drifts. This paper presents our proposed incremental GMLC method which answers the above challenges and can be applied to build a recommender system. This method is tested on the well-known MovieLens and Jester datasets, and it is able to adapt to concept drifts and maintain the Hamming loss at a low level.

✉ Khalil Laghmari
laghmari.khalil@gmail.com

Christophe Marsala
christophe.marsala@lip6.fr

Mohammed Ramdani
ramdani@fstm.ac.ma

[1] Laboratoire Informatique de Mohammedia, FSTM Hassan II University of Casablanca, BP 146, 20650 Mohammedia, Morocco

[2] Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu, Paris 75005, France

Keywords Graded multi-label classification · Recommender system · Sparse data · Incremental mode · Concept drift

1 Introduction

In graded multi-label classification (GMLC) [10], each data can be associated with multiple labels with different membership grades.

For example, in a movie web catalogue where *one-to-five* star rating corresponds to membership grades, a movie can be described by attributes such as title and production year, and associated with a label set such as:

$$Set(movie) = \{action\star\star, thriller\star, comedy\star\}.$$

GMLC is about building a classifier that learns from labelled data to predict the corresponding graded label set for unlabelled data based on descriptive attributes.

When a user ends watching an online movie, he may be asked to rank that movie, in order that other movies could be recommended to him. This recommendation is proposed by a recommender system [5].

A recommender system is a model that learns to propose for each active user the set of items that are most likely interesting to him. Learning a recommender system from user data, item data, and ratings presents three main challenges. The first challenge is the scalability since data are expected to be large and continuously growing while recommendations are desired to be immediate. The second challenge is data sparsity since users cannot rank the complete large set of items. Instead they rank a small subset of items one by one with unpredictable delays. The third challenge is the cold start problem which occurs when a new user or item is

added. Indeed it is a difficult task to give good recommendations for a new user due to the lack of information about him and his preferences. The system may also fail to recommend for users a new added item when there is a large number of already positively rated items.

Collected data for recommender systems are multi-dimensional (users and items), while usual GMLC approaches can handle only uni-dimensional data. However GMLC can be used in a recommender system following two straightforward approaches. The first approach is to discard one dimension. For instance, discard item data and keep only user data as descriptive attributes, items as target attributes, and ratings as values for target attributes, or in the same way discard user data and keep only item data and ratings. The second approach is to build two GMLC models one by discarding item data, and the other by discarding user data, and then aggregate their results to output the final recommendation.

Another challenge of building a recommender system based on GMLC is concept changes over time known as concept drifts [25]. Indeed, GMLC approaches were tested only in batch mode, while for recommender systems, data are collected in a streaming context where user preferences may change over time.

This paper is motivated by all the challenging bottlenecks of recommender systems and GMLC for data streams subject to concept drifts. The main purpose of this paper is to introduce an adapted GMLC method to concept-drifting data streams that can be used to build recommender systems.

The paper is organized as follows: in Sect. 2, we introduce the GMLC task decompositions with examples. In Sect. 3, we present a formal description for the GMLC problem and we review methods that can be used to handle it. In Sect. 4, we review incremental decision tree models which were successfully applied to data streams with concept drifts. In Sect. 5, we describe the proposed GMLC method for recommender systems. In Sect. 6, we discuss experimental results, and in Sect. 7, a conclusion and plans for future work are highlighted.

2 GMLC challenges for recommender systems

2.1 Building a GMLC classifier

Let a movie be associated with the graded label set: $GSet(movie) = \{\text{action} \star\star\star, \text{thriller} \star, \text{comedy} \star\}$. Let C be the set of movie labels: $C = \{action, thriller, comedy\}$. If stars are replaced by their corresponding normalized numbers, then the label set from the above example can be interpreted as a fuzzy set of C [65]:

$$FuzzySet(movie) = \frac{3}{5}|action + \frac{1}{5}|thriller + \frac{1}{5}|comedy.$$

Table 1 Dataset example

Movie	Graded labels
1	{Action ⋆⋆⋆, thriller ⋆, comedy ⋆}
2	{Action ⋆⋆, comedy ⋆⋆⋆}
3	{Thriller ⋆⋆, comedy ⋆}
4	{Action ⋆, thriller ⋆⋆, comedy ⋆⋆}
5	{Action ⋆⋆⋆, thriller ⋆⋆⋆, comedy ⋆⋆}

Table 2 Horizontal decomposition: two MLC tasks

Movie	Labels with a grade ≥ ⋆⋆	Labels with a grade ≥ ⋆⋆⋆
1	{Action}	{Action}
2	{Action, comedy}	{Comedy}
3	{Thriller}	
4	{Thriller, comedy}	
5	{Action, thriller, comedy}	{Action, thriller, comedy}

Table 3 Vertical decomposition: three OC tasks

Movie	Action	Comedy	Thriller
1	⋆⋆⋆	⋆	⋆
2	⋆⋆	⋆⋆⋆	
3		⋆	⋆⋆
4	⋆	⋆⋆	⋆⋆
5	⋆⋆⋆	⋆⋆	⋆⋆⋆

Table 1 presents a set of 5 movies with their corresponding graded labels. The GMLC task can be seen as a combination of a multi-label classification task (MLC) [22,23,27] and an ordinal classification task (OC) [31]. Therefore, the GMLC task can be answered by decomposing it horizontally according to membership grades to a set of MLC tasks, one for each membership grade (Table 2). The MLC task corresponding to the first membership grade can be skipped because it can be deduced from the other MLC tasks. For instance, if the membership grade is not greater or equal to two stars ⋆⋆, then it is equal to one star ⋆ (Table 2). Another way to answer the GMLC task is to decompose it vertically according to the label set to a set of OC tasks, one for each label (Table 3).

Ultimately, GMLC can be time and memory consuming. It depends on the decomposition complexity, on the base classification algorithm complexity, and on the number of generated classification sub-tasks. Hence, GMLC may encounter some limitations when dealing with large datasets or with data streams subject to concept drifts [68].

2.2 GMLC for data streams subject to concept drifts

In presence of data streams, restrictions on memory and processing time are needed. Those restrictions are even more strict for the GMLC task due to its complexity; thus, some

data stream classification approaches may not be suited for GMLC.

For instance, storing data on disk or performing multiple scans [21,40,54] should be avoided. Also, handling concept drifts by rebuilding every n instances all generated classifiers by the GMLC task, or by using ensemble methods [45,59,61] should be avoided when alternatives are available.

When labelling data does not involve any subjective criteria, concept drifts are supposed to be rare, unlike the case when labelling involves subjective criteria where concept drifts may be frequent.

An example where subjective criteria are involved is the task of labelling items (books, music, movies, products, services, etc.) using an ordinal scale ranging from *strong dislike* to *strong like*. This task is based on labeller opinions, and opinions change over time so concept drifts could occur.

2.3 GMLC for recommender systems

Typically, commercial and non-commercial websites collect large size data from user opinions and preferences [24], and then they use recommender systems [5] to make item recommendations for each active user. Three main approaches that can be combined can be used to build recommender systems [43]. The first approach is demographic-based filtering which uses user information such as age, gender, city, and country. The second approach is content-based filtering which uses item information such as price, category, and production year. The third approach is collaborative filtering [64] which uses available preferences of other users.

2.4 GMLC, data streams subject to concept drifts, and recommender systems: all challenges in one

Learning from growing user opinion or preference data is a combination of the GMLC task and the stream classification task [1,18,41] with the inherent main challenges of recommender systems.

Focusing on the scalability criterion, the GMLC decomposing strategy should involve minimal data pre-processing and result on a minimal number of sub-classification tasks, where each one can be solved using a scalable classification method.

The sparsity challenge can be handled by solving sub-classification tasks using a classifier adapted to sparse data, and the lack of descriptive attributes can be tackled using a decomposing strategy allowing to learn the maximum number of target attribute relations [38]. The drawback of learning target attribute relations is that a prediction error for one target attribute may be propagated to all related target attributes. Hence an appropriate approach for learning label relations while limiting error propagation is needed [32,33].

Most of classifiers adapted to data streams predict the majority class until a fixed or dynamic threshold of sufficient data or statistics is met, and then the classifiers are updated. The cold start challenge can be addressed by reducing this threshold, but updating the classifier based on insufficient data may cause the detection of false concept drifts.

A straightforward approach to address the GMLC task in the case of data streams is to use an incremental classifier to solve the GMLC sub-tasks. However, all known GMLC approaches were tested only on static and medium-size datasets [7,10,34], and their performance in a concept-drifting data stream context has not been investigated.

Typically in recommender systems, user data and item data are saved at the beginning of the process, and then user rating of items is collected one by one. Hence the same user instance and the same item instance can be updated multiple times. This may be problematic for conventional incremental classifiers which process each data once.

3 Graded multi-label classification

Let $X = \{x_i\}_{1\leq i\leq n}$ be a set of data, and $A = A_1 \times \ldots A_p$ be the corresponding attribute space. We denote $(x_{ij})_{1\leq j\leq p}$ the value vector of x_i. Let $C = \{c_l\}_{1\leq l\leq k}$ be the set of labels, and $M = \{m_g\}_{1\leq g\leq s}$ be the ordered set of membership grades such as $\forall g < g', m_g < m_{g'}$ where usually m_1 represents the zero membership degree, and m_s represents the full membership degree.

Let $\lambda_i : C \rightarrow M$ be the mapping that associates each label $c_l \in C$ to its membership grade for x_i, and $\{\lambda_i(c_l)|c_l\}_{1\leq l\leq k}$ be the fuzzy set of C associated with x_i.

For all $x_i \in X$ the associated label set can be represented as a vector $y_i = (y_{il})_{1\leq l\leq k}$ where $y_{il} \in M_l$ is the membership grade of c_l for x_i. $M_1 \times \cdots \times M_k$ is then called the target attribute space.

The GMLC task is to learn a classifier $H : A \rightarrow M_1 \times \cdots \times M_k$ mapping each data to the correct membership grade vector.

Two different decompositions, namely the vertical and the horizontal decompositions, have been proposed to handle the GMLC task [10]. They can be combined starting first by the vertical decomposition, then applying the horizontal decomposition, or reciprocally. The obtained combined decompositions can be equivalent depending on how the results from decomposed tasks are aggregated.

3.1 Starting from the vertical decomposition

Learning a GMLC classifier can be decomposed in a vertical way to learn a set of ordinal classifiers, one for each label: $\forall l \in [1, k]$ learn a classifier $H_l : A \rightarrow M$ to predict the membership grade of the label c_l.

The GMLC classifier $H : A \rightarrow M_1 \times \cdots \times M_k$ is then defined as: $\forall x \in A, H(x) = (H_l(x))_{1 \leq l \leq k}$.

This first-level decomposition is similar to the well-known binary relevance (BR) approach. The difference is that in BR approach there is no graded membership degrees: $M = \{0, 1\}$. The drawback of this approach is the fact that label relations are ignored. This limitation is addressed by other transformation methods from MLC to single-label classification. For instance, in the classifier chains (CC) method [51] a classifier is trained for the first target attribute $H_1 : A \rightarrow M_1$, then for all next target attributes: $\{M_l\}_{2 \leq l \leq k}$ a classifier is trained using previous target attributes as additional descriptive attributes to model target attribute dependency: $H_l : A \times M_1 \times \cdots \times M_{l-1} \rightarrow M_l$ where $A \times M_1 \times \cdots \times M_{l-1} = A^l$ is the classifier H_l corresponding new attribute space. The drawback of this decomposition approach is that only target attribute relations related to a predefined order are discovered. This limitation was addressed by the classifier treillis (CT) approach [50]. In this approach, the idea is to use a directed graph where classifiers are placed on the graph nodes according to target attribute correlation, and each classifier at a specific node adds to its descriptive attribute space target attributes from parent nodes.

Each classifier $H_l : A^l \rightarrow M_l$ should solve an ordinal classification task since the target attribute values are ordinal. Different strategies can be used to handle this task [8,44]. Probably the most simple one [15] is to decompose the task into a set of $s-1$ binary classification tasks. Each binary classifier H_{lg}, $g = 2, \ldots, s$ is trained considering target attribute values greater or equal to m_g as the positive class $'+'$, and the remaining values as the negative class $'-'$. Then the classifier H_l can be defined using different aggregation strategies such as:

$$\forall x \in A^l, H_l(x) = \max(\{m_1\} \cup \{m_g, H_{lg}(x) =' +'\}_{2 \leq g \leq s}).$$

The GMLC classifier H is then defined as:

$$\forall x \in A, H(x) = (\max(\{m_1\} \cup \{m_g, H_{lg}(x) =' +'\}_{2 \leq g \leq s}))_{1 \leq l \leq k}.$$

3.2 Starting from the horizontal decomposition

Learning a GMLC classifier can be decomposed in an horizontal way to learn a set of multi-label classifiers, one for each membership grade m_g, $g \in [2, s]$, where each classifier $H_g : A \rightarrow \mathcal{P}(C)$ ($\mathcal{P}(C)$ denotes all subsets included in C) predicts the set of associated labels according to a membership degree greater or equal to m_g.

Each classifier $H_g : A \rightarrow \mathcal{P}(C)$ should solve a MLC task. This can be achieved either using an algorithm adapted to MLC [2,35,55,60] or using a transformation method such as BR, CC or CT to decompose the MLC task into k binary classification tasks. Each binary classifier $H_{gl}, l \in [1, k]$ is trained considering instances x_i where $y_{il} \geq m_g$ as positive, and the remaining instances as negative. The classifier H_g can be defined as: $\forall x \in A, H_g(x) = \{c_l, H_{gl}(x) =' +'\}_{1 \leq l \leq k}$.

The GMLC classifier H is then defined as: $\forall x \in A$, $H(x) = (\max(\{m_1\} \cup \{m_g, H_{gl}(x) =' +'\}_{2 \leq g \leq s}))_{1 \leq l \leq k}$, which is actually the same expression obtained starting first by the vertical decomposition.

The GMLC task can be easily decomposed to $k(s-1)$ single-label binary classification tasks. This transformation has two main advantages: first it allows solving the complex GMLC problem using any off-the-shelf binary classification method, and in second it allows performing GMLC on data streams since almost all binary classification methods have been extended to streaming scenarios. One drawback of this method is the class-imbalance problem which may be encountered by binary classifiers, especially for $g = 2$ or $g = s$ if few data are considered negative and the majority is considered positive, or reciprocally.

3.3 Solving single-label classification sub-tasks

Various methods can be used to address the single-label classification task such as K-nearest-neighbours method, mixture models, support vector machine, neural networks, decision rules, and decision trees. In this paper, we focus only on decision trees because they can be intuitively interpreted, they are easy to implement, and they output predictions rapidly.

In order to be adapted to our problem, a single-label classifier has to be efficient on sparse data with a considerable amount of missing values. Also, it has to be robust against over-fitting to achieve better accuracy.

3.3.1 Handling missing values

Removing incomplete instances or attributes is the easiest way to deal with missing values, but it can lead to a considerable information loss, especially when many missing values are expected.

Replacing missing values by the most frequent value [11] is the simplest data imputation method. More sophisticated data imputation techniques use the K-nearest-neighbours [42] or the expectation maximization [13] algorithm to estimate missing values, but they can be time-consuming.

A different approach for handling missing values is used by the C4.5 decision tree algorithm [46]. The idea is to compute information gain for each attribute using only instances with available values. Instances with unknown values for the selected attribute are splitted proportionally to the split of instances with known values. At the prediction step, the same splitting strategy is used for instances with missing values until they reach leaves. Then an aggregation function is applied to output the prediction. This approach is not time-

consuming and does not discard data prematurely; thus, it may be suited for problems with time issue such as stream classification.

3.3.2 Avoiding over-fitting

Over-fitting occurs when the model perfectly fits the training data while its complexity is increasing and its generalization capability is decreasing. In the case of decision trees, the over-fitting problem can be solved using different strategies.

Random forests [6] are an ensemble strategy reducing the over-fitting effect of decision trees by building an ensemble of decision trees using randomly sub-sampled data and attributes.

Incremental reduced error pruning [17] is a post-pruning strategy. The idea is to use a training subset for growing the tree and the other subset for pruning it. The first step is growing the tree without pruning, and the second step is removing leaves until the prediction error on the pruning subset cannot be further reduced.

The minimum description length (MDL) can be used as a measure to trade-off the model complexity and the misclassification error [47]. It's a pre-pruning strategy which allows a node to be splitted only if this minimizes bits encoding the tree size and the misclassification size on training data. This method does not require keeping an extra validation subset or building an ensemble of models, and it is based on a simple measure which makes the MDL method suited for problems with time and memory issues.

3.4 Learning modes

Batch learning is the usual learning mode where first training data $(x_i, y_i) \in A \times M, i \in [1, n], M = \{m_g\}_{1 \leq g \leq s}$ are collected, and then the hypothesis $H : A \rightarrow M$ mapping values from the descriptive attribute space to values in the target attribute space is built. Finally H is used to classify unlabelled data. This mode can build good hypothesis since it has access to the full training data, but due to time and memory limitations it can not handle large datasets using commodity hardware.

Sequential learning is the mode where the training algorithm receives data one by one or chunk by chunk and continuously updates without producing the hypothesis until a stopping criteria is met. A final step making the hypothesis operational is then necessary. Sequential learning mode demands less resources, and hence it is more practical for large size data processing using commodity hardware. However, it does not allow the hypothesis use before the learning step ends.

Incremental learning is a real-time sequential learning allowing labelling new data using the current hypothesis, which is updated and produced each time a new training data is received. Incremental learning is then suited for infinite concept-drifting data streams where the hypothesis is expected to change over time.

4 Incremental decision trees

The ID4 algorithm [53] allows building ID3 decision trees [48] incrementally. It incorporates the statistical Chi-square independence test to prevent over-fitting, but it is not an efficient algorithm since it cannot handle continuous attributes, it does not guarantee that the outputted tree is similar to the one outputted by ID3, and it may discard repeatedly sub-trees in some worst cases.

The ID5 algorithm [56] differs from ID4 in concept-drift handling. Indeed, instead of discarding sub-trees, the tree structure is updated by adding new nodes or by changing the position of current nodes.

The ID5R algorithm [57] produces the same decision tree that the ID3 algorithm produces, but can be slower than ID3 in some cases depending on the performed recursive tree restructuring operations.

The ITI algorithm [58] extends ID5R by incorporating the capability of handling numeric and missing values, and inconsistent training instances. A fuzzy approach has been proposed to incrementally adapt fuzzy decision trees [39].

The very fast decision tree (VFDT) method [14] builds a Hoeffding tree using Hoeffding bounds [28]. It ensures an asymptotically identical hypothesis to the one outputted by batch learning. VFDT method does not store any data in memory and keeps only statistics of data. Its major drawback is not being adapted to time-changing concepts.

CVFDT [29] is an extension to VFDT handling concept drifts. The idea is to build a new sub-tree when the current one becomes questionable, and replace it only when it becomes less accurate than the new sub-tree.

VFDT was also extended to multi-label classification [49] following ideas from another work [3] by allowing multi-label classifiers at leaves. Reported results state that the performance of the extended version, namely multi-label Hoeffding tree is acceptable only for very large datasets. The reason behind that is the fact that Hoeffding trees must collect many instances before looking for the best split at each node.

A concept drift occurs when the prediction accuracy of the current hypothesis decreases over time due to a change in the statistical properties of the target attribute. Two hard tasks are involved by the concept drift challenge: detect and adapt to unpredictable concept drifts, and differentiate between concept drifts and noise which may cause accuracy decrease.

One of the most commonly used methods for handling concept drifts is sliding windows. The idea is to memorize a number of only recent instances and discard the oldest

instance whenever a new instance is received. The hypothesis is expected to adapt to concept drifts since it forgets the oldest instances outside the sliding window.

A concept drift can be detected by keeping a reference to the immediate previous window, and checking for significant changes in the distributions. The sliding window can be of a fixed size [63] or of a variable size [4].

Sliding window-based methods for concept drift adaptation are based on the assumption that the new instances are more relevant to describe the current distributions. However this assumption does not hold in the presence of noise. This limitation can be overcome using an instance weighting-based method. For instance, the rule preserving criterion (RP) [37] ranks instances according to the impact of deleting them on the prediction accuracy and then suggests removing instances with the lowest impact instead of the oldest instances.

Some methods called blind methods can adapt to concept drift without even detecting it or being sure it occurs. For instance, in cases where the relevance of instances decreases over time it is interesting to not forget instances prematurely and instead make them less important. This can be achieved using any decay function [12].

In some application scenarios where manual labelling for new data is possible but costly, it is desired that the classifier leaves the most difficult instances to human annotators to maximize the prediction accuracy while minimizing the manual labelling cost. A straightforward approach to answer this challenge is to use a confidence threshold so that if the prediction confidence is below the threshold the classifier abstains from predicting. The drawback of this strategy is that setting a confidence threshold is a difficult task in the case of concept drifts. The droplets algorithm [36] overcome this limitation by rejecting difficult instances without using a fixed confidence threshold.

5 An adapted GMLC approach for recommendation systems

The problem addressed in this paper is to learn a recommender system based on GMLC. The main idea of our proposed recommender system is to build two incremental GMLC models. One model H^U named *User-incremental-GMLC* is built considering users as instances, and user characteristics and item ratings as descriptive attributes. The other model H^I named *Item-incremental-GMLC* is built considering items as instances, and item characteristics and user ratings as descriptive attributes.

Let p be the number of user characteristics, and q be the number of item characteristics. Considering the example illustrated in Fig. 1, $U_1 = (U_{11}, \ldots, U_{1p})$ is a user characteristic vector, $I_2 = (I_{21}, \ldots, I_{2q})$ and $I_3 = (I_{31}, \ldots, I_{3q})$ are

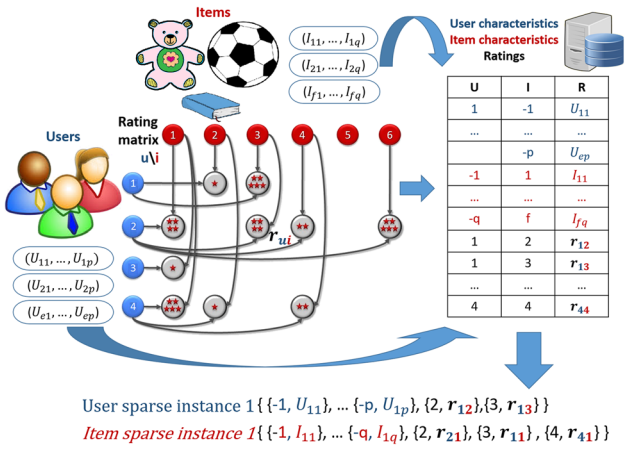

Fig. 1 Adding user and item characteristics to ratings

item characteristic vectors, r_{12} and r_{13} are the ratings given by the user U_1 to the items I_2 and I_3, respectively.

User and item characteristics can be extended over time just like ratings are added over time. For the incremental GMLC model to learn from both characteristics and ratings, they are mapped to the same dataset table as shown in Fig. 1: each row in the dataset table is defined by the user identifier, the item identifier, and the rating value. Rows corresponding to user characteristics have negative item identifiers, and rows corresponding to item characteristics have negative user identifiers. Using this mapping, the recommender system works the same way regardless whether users and items have some characteristics or not.

Since users can only rate a small subset of items, a sparse representation of user and item instances is more practical. The sparse representation used in our proposed approach is a set of identifier-value pairs preserving the order in which ratings are given (Fig. 1).

Let u be a user identifier, and u' be a user identifier corresponding to the last active user. Let $\alpha = u' - u$ be the number of last active users, and similarly let $\beta = i' - i$ be the number of last rated items.

New users, new items, and new ratings arrive continuously, but due to memory issue, only the last active users and the last rated items are kept in memory (Fig. 2). The parameters α and β should be maximized depending on the available memory capacity to minimize the number of disk access times.

When a user u gives to an item i a rating r_{ui}, the corresponding user and item instances need to be updated before feeding them to the incremental GMLC models H^U and H^I, respectively. Instances are first fetched in memory, and if they are not found, then they are retrieved from the disk. Next, the new rating value is added to the corresponding user and item instances to update the GMLC models (Fig. 3).

Each received instance is used to update the corresponding incremental GMLC model (Fig. 4). Ultimately, the same

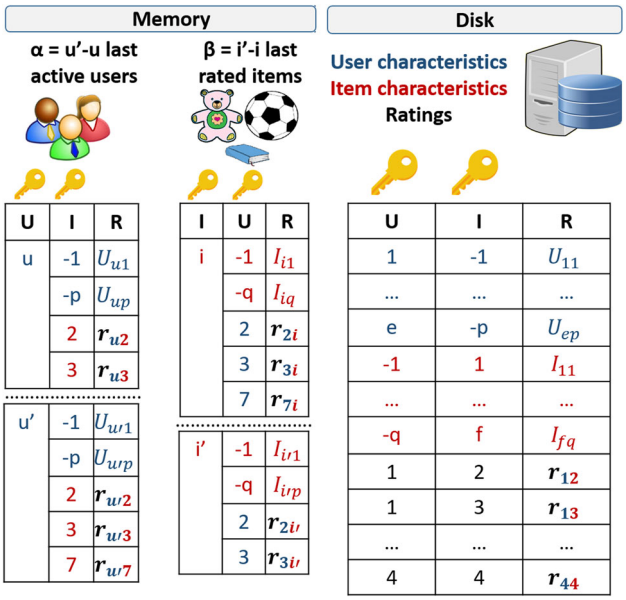

Fig. 2 Optimizing disk access

instance is received as many times as the number of ratings it has, but instances are not duplicated and only the model statistics are updated. When the number of received instances, the evaluation of the model complexity, and the evaluation of prediction errors are greater than given thresholds Δ_{inst}, Δ_{comp}, and Δ_{pred}, respectively, the model is rebuilt using the last active instances kept in memory.

5.1 Formal definition of our proposed recommender system

Let $M = \{m_g\}_{1 \le g \le s}$ be the set of possible ratings. Continuous ratings can be discretized in a pre-processing step. Let $'?'$ denotes a missing rating. For each instance x_t, the set of predicted ratings for elements not yet rated is given by: $\{H_l(x_t), x_{tl} = '?'\}_{l \ge 1}$ where each H_l is an ordinal classifier defined as: $H_l(x_t) = \max(\{m_1\} \cup \{m_g, H_{lg}(x_t) = '+'\}_{2 \le g \le s})$ where each H_{lg} is an incremental binary classifier that predicts whether the rating for the lth label is at least equal to m_g or not.

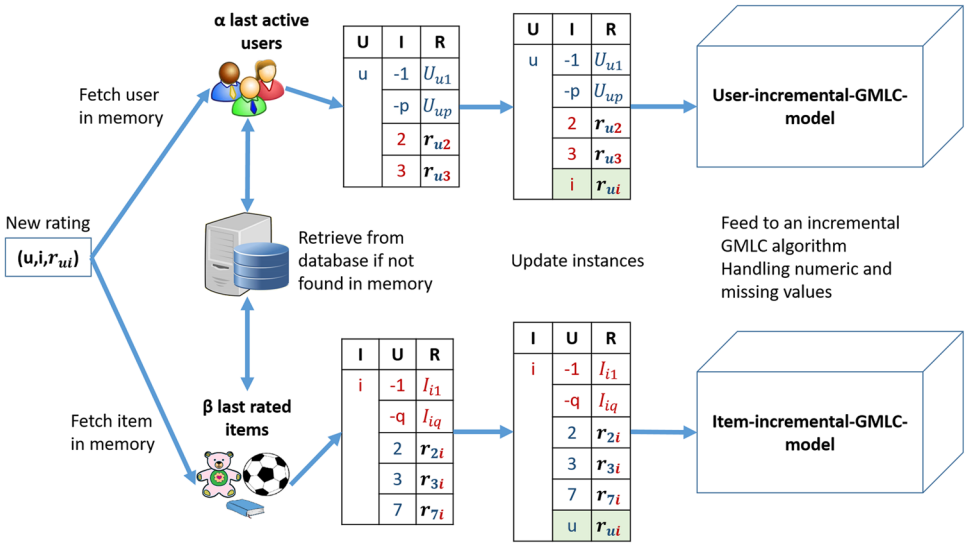

Fig. 3 Updating user and item instances

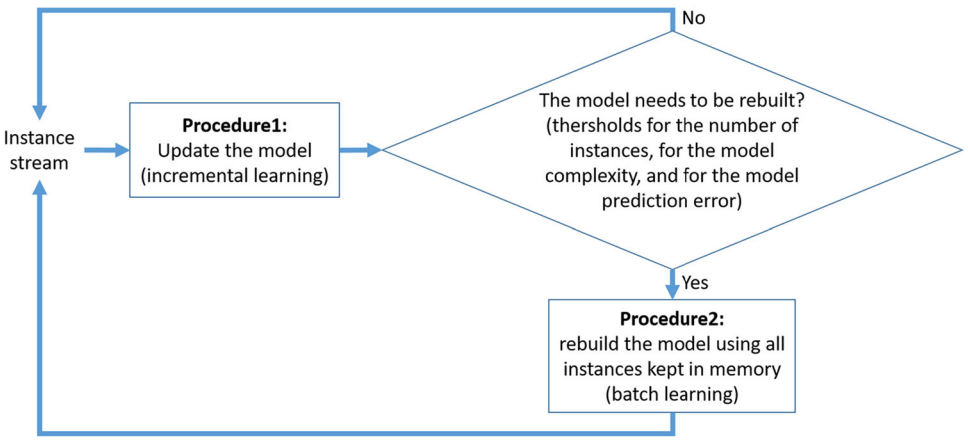

Fig. 4 Incremental learning

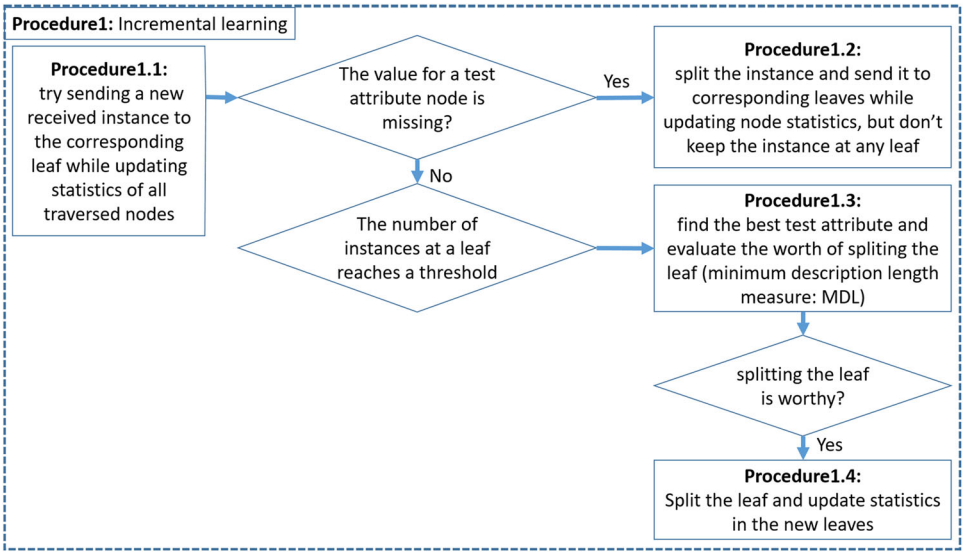

Fig. 5 The model update procedure

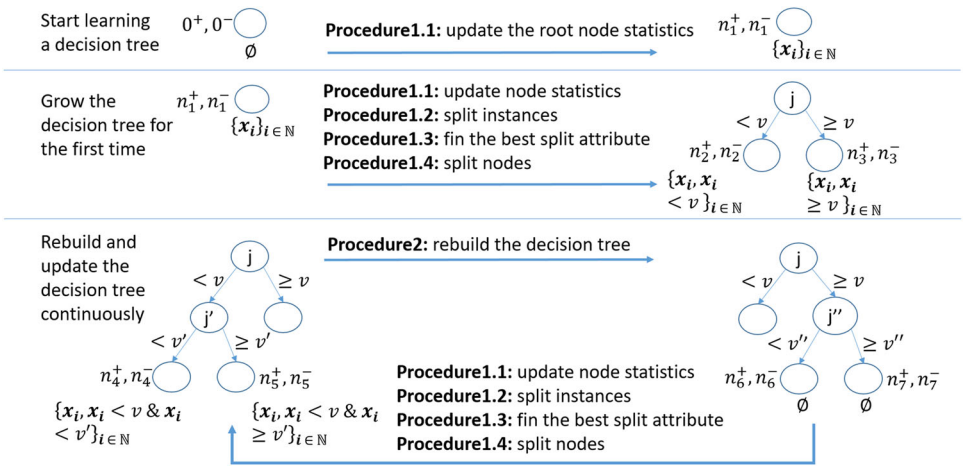

Fig. 6 Incremental decision tree

For the user-incremental-GMLC model H^U, x_t is the tth user and H_l^U predicts the rating for the lth item, while for the item-incremental-GMLC model x_t is the tth item and H_l^I predicts the rating for the lth user.

For a user instance x_u and an item instance y_i, the predicted rating given by the user x_u for the item y_i is defined by: $Agg(H_i^U(x_u), H_u^I(y_i))$ where Agg is an aggregation function typically the average or the average weighted by the confidence.

The set of aggregated predicted ratings for a user instance x_u is given by:

$$\mathbb{H}(x_u) = \{\mathbb{H}_l(x_u) = Agg(H_l^U(x_u), H_u^I(y_l)), x_{ul} = '?'\}_{l \geq 1}$$

Let η be the number of items to recommend, and $argtop_\eta$ the function that returns the first η elements having the highest predicted ratings. Our proposed recommender system R outputting the best next η items for an instance user x_u is given by:

$$R(x_u) = argtop_\eta(\mathbb{H}(x_u)).$$

5.2 Choosing the base classifier

Our proposed framework is based on $(users + items) \times (possible\ ratings - 1)$ incremental binary classifiers. The time needed to output predictions is an important criterion for the selection of the base classifier. As previously said, our proposed recommender system is based on decision trees because they are fast predictors and can be easily adapted to the streaming context of recommender systems.

For each incremental binary decision tree, a strategy similar to the one used by the C4.5 algorithm (Sect. 3.3.1) is used to handle missing values (Fig. 5: procedure 1.2). The minimum description length (MDL) principle is used as a

Table 4 Dataset information

Dataset	Model	Instances	Attributes	Labels	Label cardinality	Label density	Distinct label combinations
MovieLens	User-incremental-GMLC	6040	4	3883	165.60	0.04	6040
MovieLens	Item-incremental-GMLC	3883	3	6040	269.89	0.04	3883
Jester	User-incremental-GMLC	59,132	0	150	29.79	0.20	24,362
Jester	Item-incremental-GMLC	150	0	59,132	12,581.70	0.21	140

pre-prunning strategy to answer the over-fitting challenge (Fig. 5: procedure 1.3).

Initially the binary decision tree is only composed of a root node. Instances are added to it incrementally while updating the count of positive and negative instances (Fig. 6: procedure1.1). Each δ received instances the MDL measure is used to decide whether to split the node or not. The binary decision tree grows continuously by splitting leaf nodes until some criteria are met: first, at least Δ_{inst} instances are received. Secondly, the number of nodes in the decision tree is at least equal to Δ_{comp}. Thirdly, the Hamming loss, which is the normalized mean absolute prediction error for the last received instances is at least equal to Δ_{pred}. In practice, the correct values may not be all available, and also the decision tree may not grow enough to reach the limit size while being bad at prediction due to missing values. Hence the decision tree is rebuilt if only two of the three criteria are satisfied. Rebuilding the decision tree is done in batch mode using all instances kept in memory, but they are not kept in leaves to free memory space for new coming instances (Figs. 4, 6: procedure2).

6 Experiments

6.1 Datasets

In our experiment, we used the MovieLens dataset which involves one-to-five star ratings with user and item characteristics, and the Jester second dataset which involves continuous numeric ratings without user or item characteristics.

The MovieLens dataset [26] was released on February 2003. It is given in the form of three files: the first one contains 6040 user instances with their information: gender, age, occupation, and the zip code. The second file contains 3883 movies with their information: title, year, and corresponding categories like 'action', 'thriller', 'comedy', …. The third file contains 1,000,209 ratings ranging from 1 to 5.

The Jester dataset [24] was collected between November 2006 and May 2009. It is given in the form of 1,761,440 continuous numeric ratings of 150 jokes by 59,132 users. The host website[1] shows one joke at time, and a sliding scale of continuous values ranging from −10 to +10 to rate the joke.

Multi-label datasets are usually described using metric values such as label cardinality which is the average number of labels per instance, label density which is the cardinality divided by the number of labels, and the number of distinct label combinations. Since our approach is based on two GMLC models, those metric values are given in Table 4 for each GMLC model.

6.2 Evaluation methods for classification with data streams

Two main methods are used in the literature to evaluate classification in data streams [19,20]: the holdout error measure and the prequential error measure.

The holdout error measure E_{H} is valued on an independent test set of N instances with their corresponding target attribute values $(a_i, b_i)_{1\leq i\leq N}$. It is defined for an hypothesis $H_{[i]}$ trained on i instances and a loss function L as:

$$E_{\text{H}}(i) = \frac{1}{N}\sum_{i'=1}^{N} L(b_{i'}, H_{[i]}(a_{i'})).$$

The prequential error measure E_{P} is defined as the average of the accumulated sum of the loss function between the ground truth and the predicted value for training data:

$$E_{\text{P}}(i) = \frac{1}{i}\sum_{i'=1}^{i} L(b_{i'}, H_{[i']}(a_{i'})).$$

The value E_{P} is used as follows: each received new instance is given to the hypothesis for prediction then for training, and the loss function is computed whenever the ground truth value becomes available.

MLC evaluation functions such as precision, recall, and F-measure are not adapted to recommender systems data. They can only evaluate the ability of a classifier to predict correct labels. However, for recommender systems data it is

[1] http://eigentaste.berkeley.edu/.

not interesting to predict whether an item will be rated by a user or not, instead it is interesting to know if an item is rated, then what rating it will be given by the user. Such knowledge is what enables a recommender system to recommend only items that will be most likely highly rated by the user. Hence, loss functions measuring the distance between the predicted grade and the ground truth grade are more adapted for recommender system data.

The absolute error (AE) is a common loss function to evaluate recommender systems [16]. It is given by: $\mathrm{AE}(b_{i'}, H_{[i]}(a_{i'})) = |(b_{i'} - H_{[i]}(a_{i'})|$. The Hamming loss measure [9] can also be used to evaluate recommender systems. It is given by: $\mathrm{HL}(b_{i'}, H_{[i]}(a_{i'})) = \frac{|(b_{i'} - H_{[i]}(a_{i'})|}{m_s - m_1}$. The value $m_s - m_1$ corresponds to the maximum possible distance between the prediction $H_{[i]}(a_{i'})$ and the ground truth $b_{i'}$; hence, the Hamming loss can be interpreted as the normalized error between the prediction and the ground truth. Hamming loss is criticized in MLC because it gives too optimistic measures for data with a large number of labels, however, for recommender system data, labels correspond to membership grades supposed to be few (one-to-five star ratings for example).

6.3 Experimental set-up

The proposed recommender system approach based on GMLC has been developed from scratch using the C# language, and it has been experimented using commodity hardware of 2.2 GHz CPU speed and 4Go RAM capacity. Results are collected using $\alpha = 2000$ as the number of users stored in memory, $\beta = 2000$ as the number of items stored in memory, and $\eta = 1$ the number of items to recommend at a time.

In our experiments, the MDL measure is used to determine whether to split a leaf node or not. The MDL measure should be evaluated each time a sufficient number of instances δ are received. It has been found empirically that $\delta = 100$ to $\delta = 1000$ instances are needed before splitting a leaf to create new nodes in incremental decision trees [52]. Hence, the threshold $\delta = 100$ is used in our experiments to grow the tree faster in order to answer the cold start challenge.

Concept drifts are detected when the prediction error exceeds a threshold value Δ_{pred}. The challenge is to distinguish between true concept drifts where the decision tree should be rebuilt, and false concept drifts where it may be sufficient to transform leaves to nodes in order to improve the prediction accuracy. Rebuilding the decision tree is a costly operation and should be performed only when splitting leaf nodes does not help improving the prediction accuracy. Hence, the number Δ_{inst} of instances to receive before trying to rebuild the decision tree should be higher than the threshold δ, but not too much, otherwise concept drifts may go without being detected. The threshold Δ_{inst} should be in the range $]\delta, 2\delta[$ in order to check for concept drifts only after the worth of growing the decision tree is evaluated. Hence, the middle value $\Delta_{\mathrm{inst}} = 150$ is used in our experiments.

The Hamming loss does not exceed the value 0.3 for most of recommender systems on publicly available datasets [30, 67]. Hence, in our experiment, the model is rebuilt when the Hamming loss measure exceeds the threshold $\Delta_{\mathrm{pred}} = 0.3$.

The decision tree may grow indefinitely in case the Hamming loss does not exceed the threshold value Δ_{pred}. The decision tree is built using the last $\alpha = \beta = 2000$ instances. In case each leaf of a balanced binary decision contains exactly 100 instances the number of leaves is 20, and the number of nodes is 39. However, a threshold $\Delta_{\mathrm{comp}} > 39$ should be considered because instance distribution over leaves is generally not balanced for real data. In our experiments, the threshold value $\Delta_{\mathrm{comp}} = 50$ is used as the number of nodes needed to rebuild the decision tree.

Note that for this paper, the thresholds $\alpha = 2000$ and $\beta = 2000$ have been selected according to the available memory capacity. The threshold $\delta = 100$ has been selected to answer the cold start challenge based on recommended values by the state of the art. The threshold $\Delta_{\mathrm{pred}} = 0.3$ has been selected because most of state-of the-art approaches maintain the Hamming loss under the value 0.3. The threshold $\Delta_{\mathrm{inst}} = 150$ has been selected to rebuild the decision tree only after trying to split leaf nodes $\Delta_{\mathrm{inst}} \in]\delta, 2\delta[$. Finally, the threshold $\Delta_{\mathrm{comp}} = 50$ has been selected depending on the previous parameters to not let the decision tree grow indefinitely. However, our proposed approach could be improved using a different threshold set-up. An extensive empirical investigation of the impact of all used thresholds on the performance of our proposed approach is planned for future work.

The output rating is obtained by computing the average of the predicted value by the user-incremental-GMLC model and the predicted value by the item incremental GMLC model.

Ratings from the Jester dataset are discretized and mapped to a set of 11 possible value $M = \{-10, -8, \ldots, 8, 10\}$. The average aggregation function allows the predicted value to be in the interval $[-10, 10]$ and not necessarily in M.

The error between the predicted membership grade and the true membership grade for the MovieLens dataset ranges from 0 to $5 - 1 = 4$, while for the Jester dataset it ranges from 0 to $10 - (-10) = 20$. Hence the Hamming loss measure is more appropriate to compare results from different datasets with different graded membership scales.

6.4 Results

In order to evaluate our proposed framework, we used the prequential evaluation strategy (Sect. 6.2). The Hamming

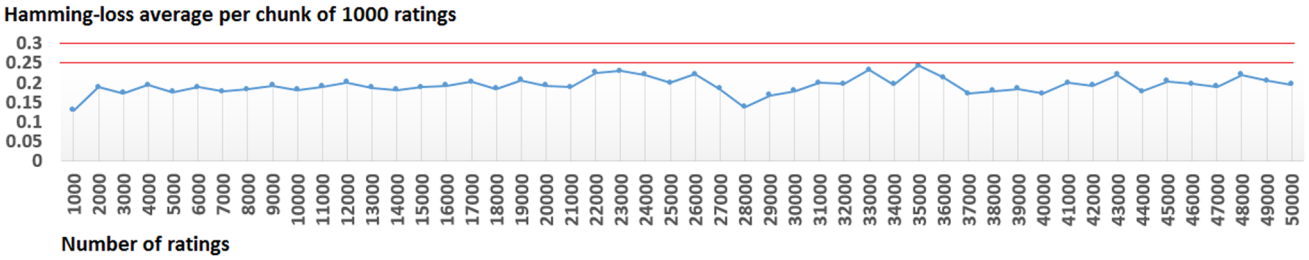

Fig. 7 Evaluation of the proposed new recommender system on MovieLens dataset

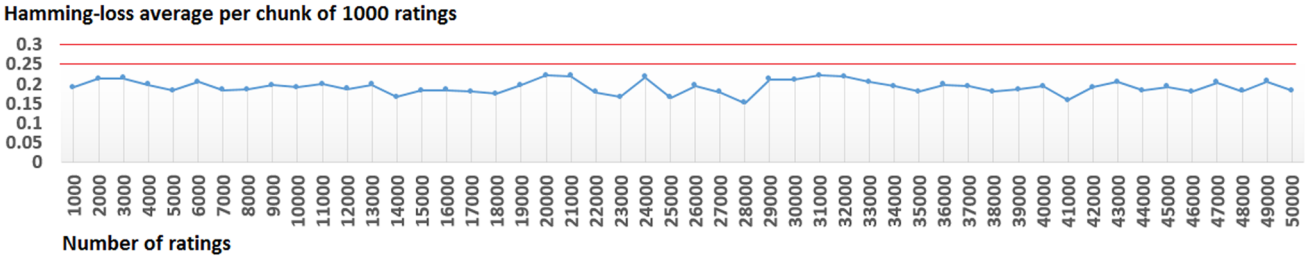

Fig. 8 Evaluation of the proposed new recommender system on Jester dataset

Table 5 Details of recommendation evaluation for MovieLens dataset

Predicted rating value	Times	Hamming loss
1	28	0.0179
1.5	109	0.1456
2	835	0.2030
2.5	2690	0.2465
3	5988	0.2059
3.5	13,724	0.2160
4	18,283	0.1717
4.5	7287	0.1852
5	1057	0.1055

Table 6 Details of recommendation evaluation for Jester dataset

Predicted rating value	Times	Hamming loss
−10	15	0.0488
−9	15	0.0881
−8	27	0.1592
−7	385	0.1458
−6	798	0.1653
−5	1205	0.1832
−4	1864	0.2135
−3	2259	0.2131
−2	2893	0.2033
−1	4058	0.1972
0	5705	0.1844
1	7868	0.1746
2	8327	0.1795
3	6697	0.1960
4	4411	0.2127
5	2685	0.2230
6	696	0.1860
7	63	0.1986
8	30	0.1351

loss measure is evaluated each 1000 received ratings. Results for the first 50,000 ratings are reported for both MovieLens (Fig. 7) and Jester (Fig. 8) datasets. The obtained Hamming loss evolution curves are similar. This can be explained by the fact that the same parameters and strategy are used for the tested datasets.

Received ratings first update statistics for decision trees. After collecting enough statistics, a decision tree may grow by splitting leaf nodes if it is worthy according to the minimum description length measure. Decision trees keep growing and fitting better the dataset while the Hamming loss is decreasing. When a concept drift occurs, the Hamming loss keep increasing until the decision tree complexity or the prediction error reaches a threshold value, then the concept drift is confirmed and the decision tree is rebuilt. As a result the Hamming loss keep decreasing until another concept drift occurs.

Since each decision tree is built for a target attribute, it receives only instances having a value for that attribute. Hence decision trees don't learn necessarily from the same instances, and therefore they are not grown or rebuilt at the same time. The used strategy ensures a maintained low level of prediction errors as shown for the MovieLens (Fig. 7) and

Table 7 Performance comparison for MovieLens dataset

Approach	Hamming loss
GMLC-based RS	**0.19**
Collaborative filtering—subset from 50 neighbours	0.28
Collaborative filtering—subset from 950 neighbours	0.22
Hybrid collaborative filtering—subset from 50 neighbours	0.27
Hybrid collaborative filtering—subset from 950 neighbours	0.21
Demographic hybrid collaborative filtering—subset from 50 neighbours	0.27
Demographic hybrid collaborative filtering—subset from 950 neighbours	**0.20**
Collaborative filtering—50 neighbours	0.23
Collaborative filtering—950 neighbours	**0.20**
Hybrid collaborative filtering—50 neighbours	0.25
Hybrid collaborative filtering—950 neighbours	**0.20**
Demographic hybrid collaborative filtering—50 neighbours	0.24
Demographic hybrid collaborative filtering—950 neighbours	**0.20**
UO-CRBMF UserBased	0.19
UO-CRBMF ItemBased	**0.18**
Hybrid CRBMF	**0.18**

Jester (Fig. 8) datasets, where the Hamming loss measure is kept always under the value 0.25.

Tables 5 and 6 show for MovieLens and Jester datasets, respectively, the number of times each rating is predicted, and the corresponding Hamming loss. New ratings ({1.5, 2.5, 3.5, 4.5} for the MovieLens dataset, and odd rating values for the Jester dataset) can be predicted because our framework combine two GMLC models.

The lowest and highest ratings are predicted much less times then middle rating values. This is explained by the used aggregation strategy of binary decision trees. Indeed, the lowest rank is predicted only if the output of all other binary decision trees is the negative class for the two GMLC models. The highest rank is predicted only if the output of the corresponding decision tree is the positive class for the two GMLC models. However, all other ratings can be predicted either by the two GMLC models or by combining their predictions.

Our framework based on decision trees ensures giving a recommendation in a constant time less than one second in the worst case. The optimistic learning time is about a half second, but it can take more depending on whether the instance to learn from is kept in memory or has to be fetched in disk, and on whether only statistics are updated, or a leave node is splitted, or the decision tree is entirely rebuilt. A walltime of one second is installed to force the decision tree to output a prediction without the processing or the rebuild being necessarily finished.

Since the Jester dataset does not have user and item information the MovieLens dataset is used to compare our approach to three base recommender system approaches [67]: the collaborative filtering approach based on ratings of similar users, the hybrid collaborative filtering based on item descriptive attributes and ratings of similar users, and the demographic hybrid collaborative filtering approach based on user and item descriptive attributes and ratings of similar users. The similarity between instances for the three approaches is computed using the Pearson correlation coefficient. Evaluations of neighbourhood sizes 50 and 950 are considered because they correspond to the worst and the best results. The prediction is either based on the entire neighbourhood, or it is based on a neighbourhood subset as follows: 20% of instances having the closest similarities and 20% of instances having the farthest similarities [66]. Our approach is also compared to a recent collaborative filtering approach named 'UO-CRBMF' [62]. It is based on neural networks and incorporates user occupation information (available for the MovieLens dataset). It has three variants named 'UO-CRBMF UserBased', 'UO-CRBMF ItemBased', and 'Hybrid UO-CRBMF'. Table 7 shows that our proposed approach referred as 'GMLC-based RS' is outperformed by the UO-CRBMF approaches, but it gives better results than the three baseline approaches according to the Hamming loss measure. The bold characters in Table 7 mark the best Hamming loss value obtained for GMLC-based RS, for demographic hybrid collaborative filtering, for collaborative filtering, for hybrid collaborative filtering, and for CRBMF approaches. Our proposed approach has two main areas for improvement: the first is changing the parameter values, and the second is changing the decomposing and the aggregation strategies of the GMLC task.

7 Conclusion and future work

In this paper, we introduced a new framework for recommender systems based on two graded multi-label classification models. One considering users as instances, and user characteristics and ratings as attributes, and the other considering items as instances, and item characteristics and ratings as attributes. Incremental decision trees are used as base classifiers to handle streaming data. Each data are considered as a set of values instead of a value vector to optimize the memory use for sparse data. This allows combining instance characteristics and ratings in the learning process. Hence our recommender system framework can be applied to any dataset regardless whether users and items have already some characteristics or may have them later.

A windowing strategy is used to decide when to grow the tree depending on the minimum description length measure, and also to handle concept drifts and decide when to rebuild the decision tree depending on the number of received instances, the number of tree nodes, and on the Hamming loss measure. A windowing strategy is also used to keep the last active users and the last rated items in memory for time optimization.

Our proposed framework is tested on two different datasets in terms of the number of users and items, and also the type and the range of rating values. The proposed framework gives similar results for the same parameters and strategy used for MovieLens and the Jester datasets. The Hamming loss measure is maintained under the value 0.25, and the prediction time was maintained under one second.

The most time-consuming process in a recommender system is disk access, especially for large datasets. This issue can be answered by using a distribution strategy or a framework for storing values to disk such as the distributed file system proposed by the well-known Hadoop framework. For future works, we plan to study the impact of changing our approach thresholds on different datasets. We plan also to investigate the impact of changing the decomposing strategy, the aggregation strategy, and also the base classifier by using another classifier type than a decision tree. A good alternative could be neural networks because they are natively adapted to streaming scenarios.

Acknowledgements This work has been partially funded by the Ministère de l'Enseignement Supérieur, de la Recherche Scientifique et de la Formation des Cadres (MESRSFC) of Morocco and the French Institute of the French Embassy in Morocco.

References

1. Aggarwal, C.C.: A survey of stream classification algorithms. In: Aggarwal, C.C. (ed.) Data Classification: Algorithms and Applications, pp. 245–274. CRC Press, Boca Raton (2014)
2. Agrawal, S., Agrawal, J., Kaur, S., Sharma, S.: A comparative study of fuzzy PSO and fuzzy SVD-based RBF neural network for multi-label classification. Neural Comput. Appl. 1–12 (2016). doi:10.1007/s00521-016-2446-x
3. Amanda, C., King, R.D.: Knowledge discovery in multi-label phenotype data. In: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery, PKDD '01, pp. 42–53. Springer, London (2001)
4. Bifet, A., Gavaldă, R.: Learning from time-changing data with adaptive windowing. In: SDM, pp. 443–448. SIAM (2007)
5. Bobadilla, J., Ortega, F., Hernando, A., GutiéRrez, A.: Recommender systems survey. Knowl. Based Syst. **46**, 109–132 (2013)
6. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
7. Brinker, C., Mencía, E.L., Fürnkranz, J.: Graded multilabel classification by pairwise comparisons. In: 2014 IEEE International Conference on Data Mining, pp. 731–736 (2014)
8. Cardoso, J.S., da Costa, J.F.P.: Learning to classify ordinal data: the data replication method. J. Mach. Learn. Res. **8**, 1393–1429 (2007)
9. Chen, C.L., Chang, C.H.: Evaluation of session-based recommendation systems for social networks. In: 2013 IEEE 13th International Conference on Data Mining Workshops, pp. 758–765 (2013)
10. Cheng, W., Dembczynski, K., Hüllermeier, E.: Graded multilabel classification: the ordinal case. In: ICML, pp. 223–230 (2010)
11. Clark, P., Niblett, T.: The CN2 induction algorithm. Mach. Learn. **3**(4), 261–283 (1989)
12. Cohen, E., Strauss, M.J.: Maintaining time-decaying stream aggregates. J. Algorithms **59**(1), 19–36 (2006)
13. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. J. R. Stat. Soc. Ser. B **39**(1), 1–38 (1977)
14. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '00, pp. 71–80. ACM, New York (2000)
15. Eibe, F., Mark, H.: A simple approach to ordinal classification. In: Proceedings of the 12th European Conference on Machine Learning, EMCL '01, pp. 145–156. Springer, London (2001)
16. Ekstrand, M.D., Riedl, J.T., Konstan, J.A.: Collaborative filtering recommender systems. Found. Trends Hum. Comput. Interact. **4**(2), 81–173 (2011)
17. Fürnkranz, J., Widmer, G.: Incremental reduced error pruning. In: Cohen W.W., Hirsh H. (eds.) Proceedings of the 11th International Conference on Machine Learning (ML-94), pp. 70–77. Morgan Kaufmann, New Brunswick (1994)
18. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: A Survey of Classification Methods in Data Streams. Springer, Boston (2007)
19. Gama, J.A., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pp. 329–338. ACM, New York (2009)
20. Gama, Ja, Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Mach. Learn. **90**(3), 317–346 (2013)
21. Gehrke, J., Ganti, V., Ramakrishnan, R., Loh, W.Y.: Boat—optimistic decision tree construction. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, SIGMOD '99, pp. 169–180. ACM, New York (1999)
22. Gibaja, E., Ventura, S.: Multi-label learning: a review of the state of the art and ongoing research. Wiley Interdiscip. Rev. Data Min. Knowl. Disc. **4**(6), 411–444 (2014)
23. Gibaja, E., Ventura, S.: A tutorial on multilabel learning. ACM Comput. Surv. **47**(3), 1–38 (2015)
24. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: a constant time collaborative filtering algorithm. Inf. Retr. **4**(2), 133–151 (2001)

25. Haque, A., Khan, L., Baron, M., Thuraisingham, B., Aggarwal, C.: Efficient handling of concept drift and concept evolution over stream data. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 481–492 (2016)
26. Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. ACM Trans. Interact. Intell. Syst. **5**(4), 1–19 (2015)
27. Herrera, F., Charte, F., Rivera, A.J., del Jesus, M.J.: Multilabel classification. Multilabel Classification Problem Analysis, Metrics and Techniques, pp. 17–31. Springer International Publishing, Switzerland (2016)
28. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**, 13–30 (1963)
29. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, pp. 97–106. ACM, New York (2001)
30. Jiang, X., Niu, Z., Guo, J., Mustafa, G., Lin, Z., Chen, B., Zhou, Q.: Novel boosting frameworks to improve the performance of collaborative filtering. In: Ong C.S., Ho T.B. (eds.) Proceedings of the 5th Asian Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 29, pp. 87–99. PMLR, Australian National University, Canberra (2013)
31. Kim, S., Kim, H., Namkoong, Y.: Ordinal classification of imbalanced data with application in emergency and disaster information services. IEEE Intell. Syst. **31**(5), 50–56 (2016)
32. Laghmari, K., Marsala, C., Ramdani, M.: Graded multi-label classification: compromise between handling label relations and limiting error propagation. In: 2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA), pp. 1–6 (2016)
33. Laghmari, K., Marsala, C., Ramdani, M.: Classification multilabels graduée: Apprendre les relations entre les labels ou limiter la propagation d'erreur ? Revue des Nouvelles Technologies de l Information, Extraction et Gestion des Connaissances, RNTI-E-33, pp. 381–386 (2017)
34. Lastra, G., Luaces, O., Bahamonde, A.: Interval prediction for graded multi-label classification. Pattern Recognit. Lett. **49**, 171–176 (2014)
35. Liu, C., Cao, L.: A Coupled k-Nearest Neighbor Algorithm for Multi-label Classification, pp. 176–187. Springer, Cham (2015)
36. Loeffel, P., Marsala, C., Detyniecki, M.: Classification with a reject option under concept drift: the droplets algorithm. In: 2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Paris, 19–21 Oct 2015, pp. 1–9. IEEE (2015)
37. Loeffel, P.X., Marsala, C., Detyniecki, M.: Memory management for data streams subject to concept drift. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, pp. 387–392 (2016)
38. Loza Mencía, E., Janssen, F.: Learning rules for multi-label classification: a stacking and a separate-and-conquer approach. Mach. Learn. **105**(1), 77–126 (2016)
39. Marsala, C.: Incremental tuning of fuzzy decision trees. In: Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), 2012 Joint 6th International Conference on, pp. 2061–2064 (2012)
40. Mehta, M., Agrawal, R., Rissanen, J.: Sliq: A fast scalable classifier for data mining. In: Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '96, pp. 18–32. Springer, London (1996)
41. Nguyen, H.L., Woon, Y.K., Ng, W.K.: A survey on data stream clustering and classification. Knowl. Inf. Syst. **45**(3), 535–569 (2015)
42. Pan, R., Yang, T., Cao, J., Lu, K., Zhang, Z.: Missing data imputation by k nearest neighbours based on grey relational structure and mutual information. Appl. Intell. **43**(3), 614–632 (2015)
43. Pazzani, M.J.: A framework for collaborative, content-based and demographic filtering. Artif. Intell. Rev. **13**(5), 393–408 (1999)
44. Qiao, X.: Learning ordinal data. Wiley Interdiscip. Rev. Comput. Stat. **7**(5), 341–346 (2015)
45. Qu, W., Zhang, Y., Zhu, J., Qiu, Q.: Mining Multi-label Concept-Drifting Data Streams Using Dynamic Classifier Ensemble. Springer, Berlin, Heidelberg (2009)
46. Quinlan, J.: Learning efficient classification procedures and their application to chess end games. In: Mitchell, R.S.M.G.C.M. (ed.) Machine Learning, pp. 463–482. Morgan Kaufmann, San Francisco (1983)
47. Quinlan, J.: The minimum description length principle and categorical theories. In: Hirsh, W.W.C. (ed.) Machine Learning Proceedings 1994, pp. 233–241. Morgan Kaufmann, San Francisco (1994)
48. Quinlan, J.R.: Induction of decision trees. Mach. Learn **1**, 81–106 (1986)
49. Read, J., Bifet, A., Holmes, G., Pfahringer, B.: Scalable and efficient multi-label classification for evolving data streams. Mach. Learn. **88**(1), 243–272 (2012)
50. Read, J., Martino, L., Olmos, P.M., Luengo, D.: Scalable multi-output label prediction: from classifier chains to classifier trellises. Pattern Recognit. **48**(6), 2096–2109 (2015)
51. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Mach. Learn. **85**(3), 333–359 (2011)
52. Salperwyck, C., Lemaire, V.: Incremental decision tree based on order statistics. In: The 2013 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2013)
53. Schlimmer, J.C., Fisher, D.: A case study of incremental concept induction. In: Proceedings of the Fifth National Conference on Artificial Intelligence, pp. 496–501 (1986)
54. Shafer, J.C., Agrawal, R., Mehta, M.: Sprint: A scalable parallel classifier for data mining. In: Proceedings of the 22th International Conference on Very Large Data Bases, VLDB '96, pp. 544–555. Morgan Kaufmann Publishers Inc., San Francisco (1996)
55. Sun, Z., Guo, Z., Jiang, M., Wang, X., Liu, C.: Research and Application of Fast Multi-label SVM Classification Algorithm Using Approximate Extreme Points. Springer, Cham (2016)
56. Utgoff, P.E.: Id5: an incremental id3. In: Laird, J.E. (ed.) ML, pp. 107–120. Morgan Kaufmann, San Francisco (1988)
57. Utgoff, P.E.: Incremental induction of decision trees. Mach. Learn. **4**(2), 161–186 (1989)
58. Utgoff, P.E.: An improved algorithm for incremental induction of decision trees. In: In Proceedings of the Eleventh International Conference on Machine Learning, pp. 318–325. Morgan Kaufmann (1994)
59. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03, pp. 226–235. ACM, New York (2003)
60. Wang, X., An, S., Shi, H., Hu, Q.: Fuzzy Rough Decision Trees for Multi-label Classification. Springer, Cham (2015)
61. Wu, Q., Tan, M., Song, H., Chen, J., Ng, M.K.: Ml-forest: a multi-label tree ensemble method for multi-label classification. IEEE Trans. Knowl. Data Eng. **28**(10), 2665–2680 (2016)
62. Xie, W., Ouyang, Y., Ouyang, J., Rong, W., Xiong, Z.: User occupation aware conditional restricted boltzmann machine based recommendation. In: 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 454–461 (2016)
63. Xioufis, E.S., Spiliopoulou, M., Tsoumakas, G., Vlahavas, I.: Dealing with concept drift and class imbalance in multi-label stream classification. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI'11, Vol. Volume Two, pp. 1583–1588. AAAI Press (2011)

64. Yang, X., Guo, Y., Liu, Y., Steck, H.: A survey of collaborative filtering based social recommender systems. Comput. Commun. **41**, 1–10 (2014)
65. Zadeh, L.: Fuzzy sets. Inf. Control **8**(3), 338–353 (1965)
66. Zaier, Z., Godin, R., Faucher, L.: Evaluating recommender systems. In: 2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution, pp. 211–217 (2008)
67. Zaier, Z., Godin, R., Faucher, L.: Recommendation quality evolution based on neighbors discrimination. In: 2008 International MCETECH Conference on e-Technologies (mcetech 2008), pp. 148–153 (2008)
68. Žliobaitė, I., Pechenizkiy, M., Gama, J.: An Overview of Concept Drift Applications. Springer, Cham (2016)