A Thesis Title

Hernan Santiago Gonzalez Toral

Department of Computer Science

University College London

hernangt12re3@gmail.com

Supervisor

Dr. Jun Wang

jun.wang.l@cs.ucl.ac.uk

This report is submitted as part requirement for the

MSc in Web Science & Big Data Analytics

at

University College London.

It is substantially the result of my own work except

where explicitly indicated in the text.

Department of Computer Science

University College London

July 20, 2016

This page is purposely left blank.

Abstract

My research is about stuff.

It begins with a study of some stuff, and then some other stuff and things.

There is a 300-word limit on your abstract.

This side is purposely left blank.

Acknowledgements

Acknowledge all the things!

This side is purposely left blank.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Overview

1.2 Dissertation Objectives and Structure

Some stuff about things.[**?**] Some more things.

Inline citation: Anne Author. Example Journal Paper Title. *Journal of Classic Examples*, 1(1):e1001745+, January 1970

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Chapter 2

Background

2.1 Recommender Systems

use background patterns of **An MDP-Based Recommender System**

2.1.1 Matrix Factorization

In a recommendation system such as Netflix or MovieLens, there is a group of users and a set of items (movies for the above two systems). Given that each users have rated some items in the system, we would like to predict how the users would rate the items that they have not yet rated, such that we can make recommendations to the users

the task of predicting the missing ratings can be considered as filling in the blanks such that the values would be consistent with the existing ratings in the matrix.

The intuition behind using matrix factorization to solve this problem is that there should be some latent features that determine how a user rates an item. Hence, if we can discover these latent features, we should be able to predict a rating with respect to a certain user and a certain item, because the features associated with the user should match with the features associated with the item.

In trying to discover the different features, we also make the assumption that the number of features would be smaller than the number of users and the number of items we have a set U of users, and a set D of items. Let $\mathbf{R}$ of size $|U| \times |D|$ be the matrix that contains all the ratings that the users have assigned to the items. Also, we assume that we would like to discover K latent features. Our task, then, is to find two matrics matrices $\mathbf{P}$ (a $|U| \times K$ matrix) and $\mathbf{Q}$ (a $|D| \times K$ matrix) such that their product approximates $\mathbf{R}$:

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q^T} = \hat{\mathbf{R}} \tag{2.1}$$

Each row of $\mathbf{P}$ would represent the strength of the associations between a user and the features. Similarly, each row of $\mathbf{Q}$ would represent the strength of the associations between an item and the features. We have to find a way to obtain $\mathbf{P}$ and $\mathbf{Q}$. One way to approach this problem is the first intialise the two matrices with some values, and calculate how 'different? their product is to $\mathbf{M}$ by using gradient descent with a regularization term in order to find the minimum difference between $\mathbf{R}$ and the predicted preference matrix $\hat{\mathbf{R}}$ (using the MSE).The update rules and the cost function are defined as follows:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e^2_{ij} = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik}) \tag{2.2}$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e^2_{ij} = q_{kj} + \alpha(2e_{ij}p_{ik} - \beta p_{kj}) \tag{2.3}$$

$$e^2_{ij} = (r_{ij} - \hat{ri}j)^2 = (r_{ij} - \sum_{k=1}^{K} p'_{ik}q'_{kj})^2 \tag{2.4}$$

2.2 Reinforcement Learning

as a reinforcement learning task called the multi-armed bandit [Sutton and Barto 1998]

Reinforcement learning involves modifying the behavior of a system such that the expected value of some possibly delayed scalar feedback is maximized.

Reinforcement learning [7, 19] is a family of machine learning algorithms that optimize sequential decision making processes based on scalar evaluations called reinforcements. The objective is to maximize the expected sum of future reinforcements for each state.

If for each state the best action is taken according to the estimated value function, then the policy is called greedy.

An MDP is a model for sequential stochastic decision problems where an autonomous agent is influencing its surrounding environment through actions

For categorizing RL agents, base the writing on the definition of approaches to approximate the optimal policy in **DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation**

Q-Learning in which values are estimated for each state and action combination

Exploration/Exploitation exploitation corresponds with the generation of trust and the exploration with the presentation of new and unexpected items

the exploitation makes the behavior of the users more predictable, making estimates of the values more reliable. The exploration makes sure that the users have a chance to visit all pages, since it allows for all alternatives to be tried

2.3 Recommender Systems and Reinforcement Learning

**References to papers are temporarily described with their title in order to facilitate the identification of the source, but they will be replaced with proper latex references

The first experiments on using reinforcement learning techniques were focused on developing recommendation systems for web content and using implicit data from server log files which stored the navigation of the users throughout their contents.

Exploration/Exploitation in Adaptive Recommender Systems considered reinforcement learning with an exploration/exploitation approach of unknown areas to automatically improve their recommendation policy for helping users to navigate through an adaptive web site. They state that a model-free algorithm like Q-Learning can become infeasible if the number of actions are so high. Additionally, they showed that a greedy policy can indeed lead to a suboptimal recommendation model as the use of such kind of policy function, where the b̈estäctions (e.g with higher probability) are taken, does not always improve the policy, so to avoid reinforcing values for states with initial high values, a random exploration process needs to be added to the action selection. The popular exploration ϵ greedy policy algorithm which modifies the action selection by selecting a greedy action with probability $(1-\epsilon)$ or an arbitrary action with probability ϵ otherwise in order to explore other existing options for the user. However, the exploration/exploitation dilemma came out as the exploration makes the future of a sequence less predictable influencing the estimation of the value function itself but inconsistent random actions have to be chosen to obtain a better policy. They finally demonstrated that this problem is solved by starting the algorithm with a lot of exploration and gradually reduce the parameter ϵ when the policy is getting closer to the optimum. Results concluded that a recommender system without exploration potentially can get trapped into a local maxima.

New Recommendation System Using Reinforcement Learning presented a general framework for web recommendation that learns directly from customer's past behavior by applying an RL process which uses the SARSA method[***ref here] and a ϵ-greedy policy. The system was composed of two models: a global model to keep track of customers trends as a whole, and a local model to record the user's individual browsing history. In general, the model treats pages as states of the system and links within a page as actions. To predict the next state, it used a ranking system that is also separated into 2 parts. i) a global ranking system using the data from a Q_{global}-matrix obtained from the action-value function to choose the next state by applying an ϵ-greedy policy. It gave the chance for rank new items that have few clicks but may match a user's interest. ii) a local ranking Q_{local} which uses an inverse ϵ-greedy policy to estimate the next state. It gave to customers more chance to explore other products than the ones they already visited. As a result, the system finds the final total reward using $Q_{total} = Q_{local} + wQ_{global}$ where $w \in (0 - 1]$ is a weight hyper parameter of the model.

Authors performed some experiments to find a relationship between ϵ and the user click rate. The findings showed that a value of $\epsilon = 0.2$ preserves the balance between exploration and exploitation, meanwhile If $\epsilon < 0.2$ the system will gave small chances to users to explore new items, or may include products that do not match to the customer's interest otherwise ($\epsilon > 0.2$). Even if the purpose of Q_{local} was to promote new products, it was less effective than Q_{global}. In conclusion, the result of this experiments explained that if a user is allowed to do much exploration, she has a chance to find non-relevant products, but if users just exploit the system, they have a chance to just keep receiving the same recommended items over time.

An MDP-Based Recommender System argue that it is more appropriate to formulate the problem of generating recommendations as a sequential optimization problem so they described a novel approach for a commercial web site based on Markov Decision Processes (MDPs) together with a novel predictive model that outperformed previous models. An MDP-based recommender system can take into account the expected utility and the long-time effect of a particular recommendation, and suggest an item whose immediate reward is lower, but leads to more *profitable* rewards in the future. Moreover, these considerations are taken into account automatically by any optimal policy generated for an MDP model which ensures a balance in the generation of the maximal expected reward stream. However, the previous benefits are offset by the fact that the model parameters are unknown, randomly initialized and that they take considerable time to converge. So, they introduced in their work a model-based system for collaborative filtering with a strong initial model, which solves quickly, and that does not consume too much memory.

Before implementing the recommender, they initialize a predictive model of user behavior using data extracted from the web site, and then use it to provide the initial parameters for the MDP. The predictive model can be thought of as a first-order Markov chain (MC)[***] of user dynamics in which states correspond to sequences of *k* events (in this case: previous selections) representing relevant information about the users interaction. They experimented with small values of *k* ($k \in [1-5]$) in order to overcome data sparsity and MDP solution complexity issues. Thus, the transition function describes the probability that a user whose *k* recent selections were $x_1, ..., x_k$ will select the item x' next. Finally, enhancements on the maximum-likelihood n-gram formulation[***] were proposed in order to find a good estimate of the transition function for the initial predictive model without suffering the problem of data sparsity and bad performance.

The proposed maximum-likelihood estimation include three major enhancements. First, a form of skipping based on the observation that the occurrence of the sequence $x_1, x_{2,x}3$ lends some likelihood to the sequence x_1, x_3. In other words, after initializing the counts of each state transition using the observed data, for any given user sequence

$x_1, x_2, ..., x_n$, they added a fractional count $1/2^{(j-(i+3))}$ to the transition from $s = \langle x_i, x_{i+1}, x_{i+2}\rangle$ to $s' = \langle x_{i+1}, x_{i+2}, x_j\rangle$, for all $i + 3 < j?n$, which acts as a diminishing probability of skipping a large number of transactions in the sequence. Equation **??** shows the updated formulation of the (normalized)maximum-likelihood estimation, where $count(s, s')$ is the fractional count associated with the transition from s to s'.

$$tr_{MC}^{base}(s, s') = \frac{count(s, s')}{\sum_{s'} count(s, s')} \tag{2.5}$$

The second enhancement exploits the similarity of sequences in a form of clustering. The idea is that the likelihood of transition from s to s' can be predicted by occurrences from t to s', where s and t are similar. Equation **??** define the similarity of states s_i and s_j where $\delta(\dot{;})$ is the Kronecker delta function and s_i^m is the m-th item in state s_i. Then, the similarity count from state s to s' was defined using equation **??**. Finally, the new transition probability from s to s' given by equation **??** yielded the best results during evaluation.

$$sim(s_i, s_j) = \sum_{m=1}^{k} \delta(s_i^m, s_j^m) \cdot (m + 1) \tag{2.6}$$

$$simcount(s, s') = \sum_{s_i} sim(s, s_i) \cdot tr_{MC}^{base}(s, s') \tag{2.7}$$

$$tr_{MC}(s, s') = \frac{1}{2} tr_{MC}^{base}(s, s') + \frac{1}{2} \frac{simcount(s, s')}{\sum_{s''} simcount(s, s'')} \tag{2.8}$$

Due to larger values of k lead to states that are more informative whereas smaller values of k lead to states with more statistical meaning, they added as a third enhancement a finite mixture modeling (e.g. combine a trigram, a bigram and a unigram into a single model) in a way to balance these conflicting properties by mixing k models, where the *i-th* model looks at the last i transactions. In addition, they applied mixture weights during experiments as $\pi_1 = \ = \pi_k = 1/k$ so the generation of the models for smaller values entails little computational overhead.

An evaluation on the accuracy of the predictive model was performed using real user transactions and browsing paths (from web logs) from the commercial store web site. Besides, they compared different variations of their enhancements on the MC approach with other models like i)both sequential and non-sequential form of the *Microsoft Commerce Server 2000*(Heckerman et al. (2000))[***] (local distributions are probabilistic decision trees); and ii)unordered versions of MC models. Results outlined that the MC models using skipping, clustering, and mixture modeling yielded better results than any other other variation of model *i*. Sequence-sensitive models outperformed the accuracy of non-sequential models, while MC models were superior to the predictor models. and the skipping enhancement was only beneficial for the transactions data set

On the other hand, the MDP-based recommender system, models the recommendation process as a sequence of states and attempts to optimize it. Here, the predictive model played an important role in the construction of the model as it provides the probability, denoted by $Pr_{pred}(x|x_1, ..., x_k)$, that a user purchase a particular item x given her sequence of past purchases $x_1, ..., x_k$. The components of the reinforcement learning model are defined as follows: a) the states are the k-tuples of items purchased; b) the actions correspond to a recommendation of an item; c) the rewards depends on the last item defining the current state only, where its net profit of the item was used; and d) the transition function is the stochastic element of the model and estimates the user's actual choice. In brief, equation **??** represents the transition function or the probability that a user will select item x'' given that item x' is recommended in state $\langle x_1, x_2, x_3 \rangle$.

$$tr^1_{MDP}(\langle x_1, x_2, x_3 \rangle, x', \langle x_2, x_3, x'' \rangle) \qquad (2.9)$$

In order to solve the MDP, the policy iteration algorithm was used as their state space enjoys of the following features that lead to fast convergence: i) the inherent directionality that transitions showed are useful to reduce the running time of the solution algorithm; ii) the computation of an optimal policy is not sensitive to variations in the number of k past transactions a state represent; iii) ignoring unobserved states by maintaining transition probabilities only for states where a transition actually occurred; and iv) using the independence of recommendations or Markov property, where the probability that a user buys a particular item depends only on her current state, allowing the algorithm to handle large action spaces.Tests performed on real data showed that the policy iteration converges after a few iterations.

To update the model, the system used an off-line approach which keeps track of the recommendations and the user selections and build a new model at fixed time intervals (e.g. once a week). So, to re-estimate the transition function at time $t+1$ the following counts are obtained:

$$c_{in}^{t+1}(s, r, s \cdot r) = c_{in}^{t}(s, r, s \cdot r) + count(s, r, s \cdot r) \tag{2.10}$$

$$c_{out}^{t+1}(s, r, s \cdot r) = c_{out}^{t}(s, r, s \cdot r) + count(s, s \cdot r) - count(s, r, s \cdot r) \tag{2.11}$$

$$c_{total}^{t+1}(s, s \cdot r) = c_{total}^{t}(s, r, s \cdot r) + count(s, s \cdot r) \tag{2.12}$$

$$tr(s, r \in R, s \cdot r) = \frac{c_{in}^{t+1}(s, r, s \cdot r)}{c_{total}^{t+1}(s, s \cdot r)} \tag{2.13}$$

$$tr(s, r \notin R, s \cdot r) = \frac{c_{out}^{t+1}(s, r, s \cdot r)}{c_{total}^{t+1}(s, s \cdot r)} \tag{2.14}$$

$$c_{in}^{0}(s, r, s \cdot r) = \xi_S \cdot tr(s, r, s \cdot r) \tag{2.15}$$

$$c_{out}^{0}(s, r, s \cdot r) = \xi_S \cdot tr(s, r, s \cdot r) \tag{2.16}$$

$$c_{total}^{0}(s, s \cdot r) = \xi_S \tag{2.17}$$

where Equation **??** corresponds to the number of times a recommendation r was accepted in state s, equation **??** is the number of times a user select item r in state s even though it was not recommended, equation

?? is the number of times a user pick item r while being in state s, regardless of whether it was recommended or not, and finally, equations **??** and **??** represent the actual probability that a user will select item r given state s when r was recommended or not respectively.

The transition function had to be carefully initialized in order to be fairly accurate when the system was first deployed to avoid the cold-start problem, so the at time $t = 0$ were calculated using equations **?? ?? ??**, where $\xi_s = 10 \cdot count(s)$, causing states that are infrequently observed to be updated faster than already observed states. Moreover, when the first transition to a state $s \cdot r$ is observed, its probability is initialized to 0.9 the probability of the most likely next item in state s with $\xi_s = 10$. In this way, the system balances the need to explore unobserved items in order to improve its model by recommending non-optimal items occasionally until getting their actual counts.

Finally, an empirical validation of their thesis compared the performance in terms of their value or utility and computational cost of the MDP-based recommender system with the performance of a recommender based on the predictive model (MC) as well as other variants. The deployed system was built using three mixture components (combined using an equal weight), with history length $k \in [1, 3]$ for both the MDP and MC model, and used the policy-iteration procedure and approximations. The average profit generated by users using the site was 28% higher for the MDP group. while in terms of computational costs, the MDP-based model provided fastest recommendations at the price of more memory use, and also built models more quickly.

All in all, the approach presented make use of off-line data to initialize a model in order to provide an adequate initial performance and overcome the cold-start problem, the authors avoid using some form of reinforcement learning technique as they argue that its implementation requires many calls and computations by a recommender system online, which will lead to slower responses, producing undesirable results for the web site owner.

Usage-Based Web Recommendations: A Reinforcement Learning Approach proposed a machine learning perspective based on a off-line reinforcement learning model to solve the recommendation problem using the Q-Learning algorithm while employing concepts and techniques commonly applied in the web usage mining domain. They argued that their decision on using this method seems appropriate for the nature of web page recommendation problem as it provides a mechanism which is constantly learning, does not need periodic updates, can be easily adapted to changes in the website structure and new trends in users behavior. Consequently, the system was trained using web usage logs available as the training set and experimental evaluations were carried out in order to demonstrate how this approach can improve the quality of web recommendations.

The RL problem formulation was considered as as a competition between different recommender systems to gather more points, than a 2-player game, as in the recommender system environment, self-play, a typical technique used in training RL systems[***], cannot be used to train the system so the system needed of actual web usage data for training. Moreover, the model, which has a stochastic nature, consists of the following properties:

states: showing the history of pages visited by the user so far. For this case, a notion of N-Grams was adopted and a sliding window of size w was set to limit the page visit sequences to a constant number and avoid large state spaces.

actions: consists of a single page recommendation at each state.

policy: computes the reward of performing an action a in state s. It rewards actions positively if it recommends a page that will be visited in one of the consequent states (not necessarily the immediate next state).

Formally, rewards are dependent on the next state, the intersection of previously recommended pages in each state and the current page sequence of the state.

As the model does not have a predetermined reward function $R(s, a)$ or a transition function $\delta(s, a)$ to give the next state, the reward can be estimated by considering each state s is formed by two sequences $V_s = \langle p^V_{s,1}, p^V_{s,2}, ..., p^V_{s,w}\rangle$, and $R_s = \langle p^R_{s,1}, p^R_{s,2}, ..., p^R_{s,n}\rangle$, indicating the sequence of visited and previously recommended pages respectively, where $p^V_{s,i}$, indicates the ith visited page in the state and $p^R_{s,i}$ indicates the ith recommended page in the state s, Additionally, they had two considerations in the implementation of the reward function: i) only the occurrence of the last page visited in the recommended pages list in state s' is used to reward the action performed in the previous sate s, and ii) the time the user spends on a page, assuming that the more time a user spends on a page the more interested. So, the reward function was defined as follows:

1. Assume $\delta(s, a) = s'$
2. $P_R = V_{s',w} \cap R_{s'}$
3. if $p \neq \emptyset$
4. For page p in P_R
    (a) $r(s, a) += reward(Dist(Rs', p), Time(p^v_w))$

where $Dist(R_i, p)$ is the distance of page p from the end of the recommended pages list and $Time(p^v_w)$ indicates the time user has spent on the last page of the state. Finally, The $reward(Dist, Time)$ function was defined as a linear combination of both values as $reward(Dist, Time) = \alpha \times dist + \beta \times Time$ with $\alpha + \beta = 1$.

Subsequently, the definition of the learning algorithm also took into consideration the following characteristics: 1) the Q-Learning algorithm given by equation **??** was proposed as an update rule and the structure to estimate how successful a prediction can be. Here, the decreasing value of α_n caused these values to gradually converge and decreases the impact of changing reward values as the training continues; 2) an ϵ-greedy action selection was picked as it is important to add some exploration of the state space especially at the beginning of the training; 3) the algorithm followed a TD(0) off-policy learning[***] procedure, as the maximum Q value of the next state is considered when estimating the future reward; and 4) the computation of $r(s,a)$ suffered a slightly change by considering value of $Q(s',a)$ with a coefficient γ in order to propagate the value of performing a specific action beyond the limits imposed by the w. As the authors mentioned in their work: "the basic idea is that when an action/recommendation is appropriate in state Si, indicating the recommended page is likely to occur in the following states, it should also be considered appropriate in state S_{i-1} and the actions in that state that frequently lead to S_i". During the training phase, the described algorithm converged after a few thousand (between 3000 and 5000) visits of each episode or user session.

$$Q_n(s,a) = [(1-\alpha_n)Q_{n-1}(s,a) + \alpha_n[r(s,a) + \gamma \max_{a'} Q_{n-1}(\delta(s,a),a')] \tag{2.18}$$

with

$$\alpha_n = \frac{1}{1 + visits_n(s,a)} \tag{2.19}$$

For the purpose of performing a set of experimental evaluations of the proposed model, simulated log files generated by a web traffic simulator were used to train and tune the rewarding functions. The metric used for each evaluation were Recommendation Accuracy, Coverage (similar to precision and recall metrics) and Shortcut Gain (measures how many page-visits users can save if they follow the recommendations). First of all, an experiment with different values of w showed that fixed window size of 3 as recommendation history resulted on better accuracy and shortcut gain, so this value was set for the rest of system evaluations. Next, they experimented the impact of gradually increasing (in steps of 5%) the coefficient α of parameter *Dist* in the reward function, resulting on both higher accuracy and higher shortcut gain for values up to 15%. This matched the natural consequence of adding a bounded size of window on recommendations history. Finally, a set of experiments also tested the system performance by increasing the coefficient γ in the reward function, obtaining an increase of the accuracy until reaching an upper bound (around 0.20%) where it began to drop, while the shortcut gain increased steadily up to a point where recommendations became so inaccurate.

Briefly, the proposed off-line reinforcement learning method used a simple formulation of the recommendation problem but it obtained much better results compared with two baselines methods: association rules and item-based collaborative filtering (with probabilistic similarity measure). As the coverage increased, naturally accuracy decreased in all systems, but the RL approach outperformed the other two systems displaying a lower rate in which its accuracy decreased. Authors concluded that the algorithm is a good candidate for solving the recommendation problem as it does not rely on any previous assumptions regarding the probability distribution of visiting a page after having visited a sequence of pages, and that the nature of the problem matches perfectly with the notion of delayed reward (known as temporal difference). In addition, they suggested that it could be possible to use a more complicated formulation of the reward function rather than a linear combination of factors, such as a neural networks, in order to produce better results.

Later, in **A Hybrid Web Recommender System Based on Q-Learning** they exploited the previous reinforcement learning framework and present a hybrid web recommendation method enriched with semantic knowledge about the usage behavior as a way to improve the RL solution and obtain a more generalized solution regarding to the usage data it has. The new system used the incremental DCC[***] clustering algorithm to map pages to higher level concepts, and thus exploit the hierarchical and conceptual document clustering to provide a semantic relationship between them in combination with the usage data in a user session. Therefore, each state now consists of a sequence of concepts visited by the user, while actions are recommendation of pages that belong to a specific concept. This definition resulted in a much smaller state-action space as the size of the state space is now dependent on the number of distinct page clusters.

Now, an action a recommending a concept c is rewarded if the user visits a page belonging to concept c later in his browsing session. The new reward function shown in equation **??** takes into account the content similarity of the recommended and visited pages, where CBR represents the content-based reward of an action (which is equal to the similarity score between concepts) defined in equation **??**, and UBR is the usage-based reward defined in the previous approach.

$$UBR(Dist(R_{s'}, r), Time(P_{t+1})) \times CBR(r, C(P_{t+1})) \tag{2.20}$$

$$l(c) = -log_{\surd}(c) \tag{2.21}$$

$$Sim(c_1, c_2) = max_{a \in LCA} l(a) \tag{2.22}$$

To sum up, the algorithm did not make predictions based on weak usage patterns as now the states represent a generalized view of many single visit sequences. However, this approach seems specifically appropriate for the off-line training phase. On the other hand, evaluation results showed the flexibility of the new RL approach to incorporate different sources of information to improve the quality of recommendations, and also demonstrated how it could be extended in order to incorporate various sources of information.

Improving adaptation of ubiquitous recommender systems by using reinforcement learning and collaborative filtering proposed a learning system for Context-based Recommender System (CBRS)[***] by modeling an MDP agent which combines a hybrid Q-learning algorithm (HQL), collaborative filtering and case-based reasoning techniques in order to define a *contextual* recommendation process based on different context dimensions (cognitive, social, temporal, geographic). It addressed the following problems that came out in recommender systems: a)avoid the intervention of experts as the use of the Q-learning algorithm does not need initial user?s information; b)reduce the cold start problem thanks to the ability of the Q-learning algorithm to explore the knowledge of other users in the same context by using CF; c)accelerate the learning process by mixing Q-learning with case-based reasoning techniques (reuse of cases yields to faster user satisfaction); and d)an exploration strategy allows recommendations to adapt to the user's interest evolution.

As a result, the model was based in three main algorithms. First, a hybrid-based CF approach which combines the advantages of the memory-based (fill the missing rating values of the user-item matrix) and model-based CF (form the nearest neighbors of each item). Second, the Case based reasoning (CBR)[***] algorithm was picked as it uses knowledge of previous cases to solve new problems, by finding a similar past case and thus reusing it to solve the current situation. Finally, the Q-learning algorithm was improved by: i)reusing past cases information gathered from the CBR, and ii)giving the ability to use information from other users sharing the same interests, by extending the ϵ-greedy strategy to select a random action based on the similarity of user profiles (obtained from the CF algorithm).

The global mechanism of a context-based recommender system (CBRS) which uses the HyQL algorithm was composed mainly by the following modules:

sensing module: detects time, location, cognitive and social dimensions of the context.

thinking module: composed by an abstraction phase that is based on inference rules defined on temporal and space ontologies, and an aggregation phase the two dimensions.

reasoning module: chooses an action based on the HyQL algorithm

Finally, evaluations comparing the Q-learning and HyQL with respect to solving the cold start problem were performed, and consisted on testing the precision of the first 100 trials of the system starting when the user is connected to the system. In General, results showed that the precision of HyQL was greater than the precision of Q-learning, demonstrating that this approach is a good candidate to be used in the recommender system problem. However, as the system implementation depends on the context data, it needs of a hand-crafted feature creation process in order to be used on different situations.

Exploration in Interactive Personalized Music Recommendation: A Reinforcement Learning Approach formulated an interactive and personalized music recommendation approach as a reinforcement learning task based on the multi-armed bandit problem, where songs were treated as arms and user ratings as payoffs, and the objective function was set to maximize the cumulative reward of a targeted user over the long term. The model considered the exploration-exploitation tradeoff and a Bayesian model in order to deal with both audio content and the novelty of recommendations and thus learn the user musical preferences online based on their feedback.

The model differs from other approaches in three aspects: i)it is based on audio content; ii)the model is highly efficient as it allows easy online updates; and iii)the model is evaluated based on online real-life user interaction data and does not need a test dataset. Moreover, each recommendation serves two objectives: (1) satisfy the user?s current musical taste, and (2) obtain user feedback needed to improve future recommendations. Even though authors mentioned that an MDP formulation can model a broader range of problems than the multi-armed bandit, it requires much more data to train and is often more computationally expensive.

Their choice to work with a the content-based approach rather CF was justified for the following reasons. First, due to the bayesian nature of their formulation, methods for matrix factorization (MF) are much more complicated than the proposed linear model. Second, MF often needs of large amount of data for training. Third, existing Bayesian MF methods are inefficient for online updating, so they do not fit in a bandit model which is updated once a new rating is obtained, Fourth, a content-based method does not suffer of the new song problem. Finally, content-based approaches capture causality within music content rather than pure correlation that CF methods offer.

Subsequently, the interactive, personalized recommender system used a reinforcement learning task called the multi-armed bandit[***] which addressed both the exploration-exploitation trade-off and playlist generation with a single unified model. Whereas some RL models considered only a greedy strategy that does not actively seek fort user feedback, resulting in suboptimal recommendations over the long term, this model takes into account the uncertainty of both the mean and the variance of the rating distribution, allowing the recommender to explore user preferences actively rather than merely exploiting the available rating information. An approach better than the typical ϵ-greedy method to solve of the multi-armed bandit problem, is the use of the Upper Confidence Bound (UCB)[***] algorithm, but it requires an explicit form of the confidence bound that is difficult to derive in the case of music recommendation.

The personalized music rating model was focused on audio content and novelty. The former considers the overall preference of a song and is represented by the linear function $U_c = \theta' \mathrm{x}$ where x is the feature vector of the audio content and θ is the user preference in different music features (it was considered constant over time). On the other hand, the novelty factor was defined after examining the song's repetition distribution of 1000 users? listening histories collected from the Last.fm dataset. Results showed that most of the songs a user listens to are repeats and that the frequency distribution approximately follows the Zipf?s law[***], where only a small set of songs are repeated most of the time. As a consequence, it was assumed that the novelty of a song decays immediately after it is listened to and then gradually recovers according to the function $U_n = 1 - e^{-t/s}$, where s is a recovery speed (unknown) parameter learned from user interactions and $e^{-t/s}$ is the well-established forgetting curve[***] that measures user's memory retention of a song. In other words, novel songs are those of which that at a certain time a user has little or no memory. It is important to emphasize that the definition of this factor could be different or not applicable to other recommendation contexts than the musical environment.

Overall, the rating model, defined in equation **??**, assumed that a rating is formed as the combination of the user?s preference of the song?s content and the dynamically changing novelty. Hence, each user was represented by a set of parameters $\Omega = \{\theta, s\}$, where Ω needs to be estimated from historical data, so uncertainty had to be taken into account.

$$U = U_c U_n = \theta' \mathbf{x}(1 - e^{-t/s}) \tag{2.23}$$

With this in mind, the new problem formulation was solved using the Bayesian Upper Confidence Bound (Bayes-UCB)[***] algorithm. In The Bayes-UCB, the true expected payoff U_i defined in equation **??** for arm

i is treated as a random variable and the posterior distribution $p(U_i|\mathcal{D})$ of U_i given the history of payoffs $\mathcal{D}_\updownarrow = \{(\mathbf{x_i}, t_i, r_i)\}_{i=1}^{l}$ is predicted using equations**??** and **??**. Finally, the Bayes-UCB recommends song k^* that maximizes the quantile function $argmax_{k=1...|S|}Q(\alpha, P(U_k|D_l))$ where Q satisfies$\mathcal{P}[Uk \leq Q(\alpha, P(U_k|D_l))] = \alpha$ and $\alpha = 1 - \frac{1}{l+1}$

$$\mathcal{E}[R_i] = U_i = \theta' \mathbf{x_i}(1 - e^{-t_i/s}) \quad (2.24)$$

$$p(\mathbf{\Omega}|D_l) \propto p(\mathbf{\Omega})p(\mathbf{\Omega}|D_l) \quad (2.25)$$

$$p(U_k|D_l) = \int p(U_k|\mathbf{\Omega})p(\mathbf{\Omega}|D_l)\mathbf{d\Omega} \quad (2.26)$$

Since equation **??** does not have a closed-form solution, a Markov Chain Monte Carlo (MCMC)[***] should be used as an approximate inference algorithm. However, authors argued that its performance is very low and users could wait for up to a minute until the MC converges. Hence, they developed a Bayesian model using a piecewise-linear function approximation, as well as a variational inference algorithm to approximate posterior distribution of Ω. For more details about the model definition, please refer to section 4.2 in [***](this paper). Authors mentioned that even if the model just considered audio content and novelty of music on its definition, other factors like diversity, mood or genre could be also approximated by linear functions and then added to it. All in all, the defined approximate Bayesian model was used to obtain the fixed-level quantile of $p(U_i|\mathcal{D})$ and thus estimate the expected payoff and confidence bound of the conceptual Bayes-UCB.

Evaluations of both efficiency and effectiveness were carried out over six recommendation algorithms and models (Random, LinUCB-C (Content-based), LinUCB-CN (Content and novelty based), Bayes-UCB-CN, Bayes-UCB-CN-V (Variational Inference), and Greedy-CN) demonstrating that the proposed approaches were accurate and highly efficient. The effectiveness study used the *regret*[***] metric to compare the algorithms, showing that the Bayes-UCB-based algorithms performed better than Greedy-CN due to the balance of exploration and exploitation it offers, whereas the good performance of the Bayes-UCB- CN-V indicated that the piecewise-linear approximation and variational inference were appropriately defined. Moreover, results in the efficiency study empirically proved that the developed variational inference algorithm was 100 times faster than the MCMC. Additionally, a user study and an overall evaluation of recommendation performance dropped good conclusions. Results showed that the bandit approach with the novelty factor addressed the cold-start problem improving the recommendation performance.

To sum up, this work was considered by the authors as the first to balance exploration and exploitation based on reinforcement learning and the multi-armed bandit, and thus improve recommendation performance by addressing the cold-start problem in music recommendation without relying on additional (contextual information). An approximation to the rating model and the new probabilistic inference algorithms helped to achieve real-time recommendation performance that could be generalized to other recommenders and/or media types. Finally, authors suggested that this work could be extended to model the correlations between different users to further reduce the amount of exploration by using hierarchical Bayesian models. On the other hand, if people prefer to boost the performance of CF, they suggested to exploit the exploration/exploitation trade-off idea by using the latent features learned during the matrix factorization rather than the audio features and keep other parts of the proposed system unchanged.

ENHANCING COLLABORATIVE FILTERING MUSIC RECOMMENDATION BY BALANCING EXPLORATION AND EXPLOITATION extended the work presented above by introducing exploration into the CF context. The approach used a Bayesian graphical model that takes into account the CF latent factors and novelty of recommendation, as well as a Bayesian inference algorithm to efficiently estimate the posterior rating distributions.

Authors argued that their previous work, based on a content-based approach, had experience the following drawbacks: (i) the personalized user rating model suffer of a semantic meaning between low-level audio features and high-level user preferences; (ii) it is difficult to determine which acoustic features are actually effective in the music recommendation scenario; and (iii) recommendation of songs under this approach lacks of variety due to most of them are acoustically similar. Therefore, they presented an approach based ....

Recalling the definition of Matrix factorization in Section [***], this method characterizes users and songs by vectors of latent factors $\mathbf{u_i}$ and $\mathbf{v_j}$ respectively with $i \in [1, m], j \in [1, n]$. In order to learn the latent feature vectors, the system used equation **??** and Alternating Least Squares (ALS)[***] to minimize the regularized cost function on the training set:

$$\sum_{(i,j)\in I} (r_{ij} - \mathbf{u_i^T v_j})^2 + \lambda(\sum_{i=1}^{m} n_{ui}\|\mathbf{u_i}\|^2 + \sum_{j=1}^{n} n_{vi}\|\mathbf{v_j}\|^2) \tag{2.27}$$

where I is the index set of all known ratings, λ a regularization parameter, n_{ui} the number of ratings by user i, and n_{vj} the number of ratings of song j. However, the traditional CF approach often fails to take into consideration novelty and works greedily.

As a result, a reinforcement learning approach for CF-based music recommendation based on the n-arm bandit problem was proposed. The user rating model considered that song?s rating is affected by two factors: *CF score*, (how much a user likes the song in terms of each CF latent factor) denoted by U_{CF} and defined inn terms of matrix factorization, and *novelty score* (the dynamically changing novelty of the song) denoted by U_N. Thus, the final user rating model, given in equation **??**, can be defined as a combination of the two scores, where vector θ indicates the user?s preferences for different CF latent factors, $\mathbf{v}$ is the song feature vector learned by the ALS CF algorithm, t is the time elapsed since when the song was last heard, s the relative strength of the user?s memory, and $e^{?t/s}$ the well-known forgetting curve.

$$U = U_{CF}U_N = (\theta^{\mathbf{T}}\mathbf{v})(1 - e^{-t/s}) \tag{2.28}$$

In the same way as in [***](previous paper), each user was associated with a pair of parameters $\Omega = (\theta, s)$ to be learned from the user's rating history, and their underlying ratings were considered random rather than fixed numbers (estimated using $\mathcal{E}[R_j] = U_j$). Exploration was also introduced by using both a similar Bayesian Upper Confidence Bound (UCB) sampling algorithm and Bayesian Graphical model to estimate the posterior distribution $p(U_j|\mathcal{D})$ of U_j given the target user?s rating history $\mathcal{D}$, but in terms of the CF latent factors rather than audio content. Then, the song with the highest fixed-level *quantile* value (equation **??**) of $p(U_j|\mathcal{D})$ will be recommended to the target user.

However, the efficiency of the convergence in the Bayesian inference of this approach was improved by developing a specific Gibbs sampling algorithm[***] as in this case it is simple to sample the ratings from the conditional distribution of θ, while, a Metropolis-Hastings (MH) algorithm[***] was used to draw samples of s. For a more detail explanation of the algorithms, please refer to sections 3.2 and 3.3 in [***](this paper).

Efficiency experiments were carried out over to compare the developed sampling implementation and an MCMC algorithm developed in JAGS[1]. The results showed that their proposed Gibbs sampling algorithm is hundreds of times faster than MCMC evidencing the suitability of the algorithm for online recommender systems. Additionally, an online user study which compared the effectiveness of the proposed Bayes-UCB-CF algorithm against the traditional greedy method and the previous im-

[1]`http://mcmc-jags.sourceforge.net/`

plementation Bayes-UCB-Content[***](previous paper), proved that the cumulative average rating of new algorithm significantly outperformed the baselines. All in all, the Bayes-UCB-CF algorithm achieved a better balanced exploration/exploitation trade-off and significantly showed an improvement on its recommendation performance.

In conclusion, this first attempt to remedy the greedy nature of CF approaches could enhance the performance of CF-based music recommendation significantly. Moreover, the reinforcement learning model seems to be applicable to other contexts different from the music environment, as well as it could be deployed together with the content-based RL model from [***](previous paper) in order to build a hybrid framework which combines the strengths of both approaches.

Generating Music Playlists with Hierarchical Clustering and Q-Learning

DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation considered a music recommendation framework for recommending song sequences using a reinforcement learning approach which models the preferences of both songs and song transitions as a MDP. The model-free agent demonstrated to be potentially effective in the domain and particularly good to solve the cold-start problem as it is able to generate personalized song sequences within a single listening session of 25-50 songs and with no prior knowledge of the new user's preferences.

Assuming a finite set of n musical tracks $\mathcal{M} = \{a_1, a_2, ..., a_n\}$ and playlists of length k, the adaptive playlist generation problem is defined as an episodic MDP $\langle S, A, P, R, T \rangle$ with the following components:

- a state space S composed by the ordered sequence of songs played, $S = \{(a_1, a_2, ..., a_i)|1 \leq i \leq k; \forall j \leq i, a_j \in \mathcal{M}\}$
- an actions space A formed by all possible candidates for being the next song to play, $a_k \in A$ which means $A = \mathcal{M}$

- a deterministic transition function $P(s, a) = s'$ which indicates the existing transitions from state s to s' by taking action a
- an utiliy function R(s, a) derived from hearing song a when in state s
- $T = \{(a_1, a_2, ..., m_k)\}$: the set of playlists of length k.

In order to find an optimal policy $\pi*$ which obtains the most pleasing sequence of songs to the listener, a model-based approach was chosen arguing that even though model-free approaches learn the value of taking an action a from state s directly, it requires a lot of data to converge, and it is considerably scarce in the domain music recommendation (as well as in other kind of applications). On the other hand, model-based formulations often requires a lot of computation to find an approximate solution to the MDP, but the trade-off of computational expense for data efficiency makes the model-based approach a good option for this problem.

Therefore, since P is deterministic, only the listener?s reward function R required to be modeled. It was compactly represented as the sum of two distinct components: 1) the listener?s preference over songs, $R_s : A \rightarrow \mathcal{R}$ and 2) her preference over transitions from songs played to a new song, $R_t : S \times A \rightarrow \mathcal{R}$. Hence, R was defined using equation **??**.

$$R(s, a) = R_s(a) + R_t(s, a) \tag{2.29}$$

In essence, the model represent each song as a vector of spectral auditory descriptors, however, the framework is in principle robust and agnostic to the choice of a specific song corpus. First, the listener reward (linear) function over songs R_s generates a binary feature vector using a sparse encoding of the song descriptors as $R_s(a) = \phi_s(u) \cdot \theta_s(a)$, where $\phi_s(u)$ represents the listener pleasure for a particular set of active features. Then, the listener reward (linear) function over transitions R_t

obtains a sparse binary feature vector $R_t(a_i, a_j) = \phi_t(u) \cdot \theta_t(a_i, a_j)$, where $\phi_t(u)$ is a user-dependent weight vector and θ_t is a binary feature vector, with both representing transitions between 10-percentile bins of the song descriptors they share (for learnability purposes). An evaluation on the expressiveness of feature representation showed that different sequences are indeed distinguishable. As a result, the compact representation of songs was still rich enough to capture meaningful differences in user's preferences, and the system was able to leverage knowledge using a few transition examples to plan a future sequence of songs.

On the other hand, the agent architecture is composed by two submodules: a module for learning the listener parameters (ϕ_s and ϕ_t) which performs the initialization and learning on the fly processes; and a module for planning a sequence of songs and thus estimate the next appropriate song to play. The initialization is divided in two parts: 1) the initialization of the song preferences polls the listener for her k_s favorite songs in the database and then update the value $\phi_s(u)$ using the feature descriptors of each of the selected items; and 2) the initialization of the transition preferences based on presenting different possible transitions that encapsulate the variety in the dataset, and directly asking which of a possible set of options the listener would prefer. Updates to the user feature vector $\phi_t(u)$ are carried out in the same manner as the initialization of song preferences.

After initialization, the agent begins playing songs for the listener, waiting for her feedback, and updating ϕ_s and ϕ_t accordingly. The latter is computed by using a single unified reward signal that considers the relative contributions of the song and transition rewards to set weights (equations **??** and **??**) for credit assignment (equations **??** and **??**). In general, the learning of the fly procedure is perceived as a temporal-difference update with an attenuating learning rate that balances the trust between the previous history of observations and the newly obtained signal.

$$w_s = \frac{R_s(a_i)}{R_s(a_i) + R_t a_{i-1}, a_i} \tag{2.30}$$

$$w_t = \frac{R_t(a_{i-1}, a_i)}{R_s(a_i) + R_t(a_{i-1}, a_i)} \tag{2.31}$$

$$\phi_s = \frac{i}{i+1} \cdot \phi_s + \frac{i}{i+1} \cdot \theta_s \cdot w_s \cdot r_{incr} \tag{2.32}$$

$$\phi_t = \frac{i}{i+1} \cdot \phi_t + \frac{i}{i+1} \cdot \theta_t \cdot w_t \cdot r_{incr} \tag{2.33}$$

After determining the MDP reward function, a Monte Carlo Tree Search[***] (MCTS) heuristic is used for planning. The process chooses a subset of 50 % of the songs in the database with highest R_s score, and then at each point, it simulates a trajectory of future songs selected at random (instead of asking the user her top-k songs) and calculate the expected payoff of the song trajectory using R_s and R_t. This iterative process ends when it finds the trajectory which yields to the highest expected payoff. Finally, the first item of this trajectory is selected to be the next song.

However, a re-planning must run at every step as the system actively adjusts ϕ_s and ϕ_t online based on user feedback. It could arise a problem of high complexity when the song space is very large. Therefore, to mitigate the problem, the agent exploits the structure of the song space by clustering songs according to song types. The solution is implemented using the canonical k-means algorithm (but any clustering algorithm could work) and results on a radical reduction of search complexity.

An evaluation of the cumulative reward distribution of the proposed approach against two baselines (a random agent and a greedy agent) showed that overall, the DJ-MC agent outperforms both random and greedy agents, while in terms of adding the transition reward preferences into the model, the new system presented a small but significant boost in performance compared to a model based only on reasoning about song preferences. All in all, this approach demonstrated to be a good improvement as a reinforcement learning model for music recommendation, as it enables learning from relatively few examples, and the representation captures enough of real peoples? transition reward to provide quality of recommendation sequences.

Hybrid Collaborative Filtering with Neural Networks

2.4 Deep Reinforcement Learning

Human-level control through deep reinforcement learning

(2.34)

Chapter 3

Models

3.1 Naïve Model using Matrix Factorization

3.1.1 Reinforcement Learning model

The model is defined as a Model-free approach which its main objective is to find the optimal policy which maximizes the total future reward

State space: is the set of items recommended to the user

Action space: are the single items recommended at each timestep

Value function: rating feedback received after giving a recommendation.

At current iteration h+1, we have gathered h observed recommendation history Dh = (vi, ti, ri)h i=1. Collect user rating rh and update p(? | Dh)

Chapter 4

General Conclusions

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Appendix A

An Appendix About Stuff

(stuff)

Appendix B

Another Appendix About Things

(things)

Appendix C

Colophon

This is a description of the tools you used to make your thesis. It helps people make future documents, reminds you, and looks good.

(example) This document was set in the Times Roman typeface using LaTeX and BibTeX, composed with a text editor.

Bibliography

[1] Anne Author. Example Journal Paper Title. *Journal of Classic Examples*, 1(1):e1001745+, January 1970.