ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ, ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ

Causal Transfer Learning for Personalized Recommendation Systems

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Πολυξένης Ειρήνης Κόντη

Επιβλέποντες: Γεώργιος Στάμου, Καθηγητής Ε.Μ.Π.
Μιχάλης Ζαβλανός, Καθηγητής Duke.

Αθήνα, Ιούλιος 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ, ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ

Causal Transfer Learning for Personalized Recommendation Systems

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Πολυξένης Ειρήνης Κόντη

Επιβλέποντες: Γεώργιος Στάμου, Καθηγητής Ε.Μ.Π.
Μιχάλης Ζαβλανός, Καθηγητής Duke.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 13η Ιουλίου 2022.

.....	.....	.....
Γεώργιος Στάμου	Μιχάλης Ζαβλανός	Αθανάσιος Βουλόδημος
Καθηγητής Ε.Μ.Π	Καθηγητής Duke	Επίκουρος Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2022.

.....

Πολυξένη Ειρήνη Κόντη
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σε αυτή την εργασία μελετάμε ένα πρόβλημα μεταφοράς μάθησης για εξατομικευμένα συστήματα συστάσεων, που θα επιτρέπει σε διαφορετικές ομάδες χρηστών να μαθαίνουν συνεργατικά τις προτιμήσεις τους χωρίς να μοιράζονται ιδιωτικές πληροφορίες μεταξύ τους. Συγκεκριμένα, θεωρούμε έναν server που έχει πρόσβαση μόνο σε δεδομένα που προκύπτουν από την αλληλεπίδραση των χρηστών με το σύστημα και όχι από τα υπόλοιπα προσωπικά τους χαρακτηριστικά. Μοντελοποιούμε αυτό το σύστημα ως ένα Multi-armed bandit πρόβλημα (MAB) με ξεχωριστές μηχανές συστάσεως για κάθε ομάδα χρηστών. Στόχος του συστήματος είναι να προσδιορίσει τις προτιμήσεις της κάθε ομάδας χρηστών, να κατατάξει τους χρήστες στις ομάδες αυτές, και ανάλογα με την κατάταξη αυτή να τους κάνει τις κατάλληλες συστάσεις ώστε να μεγιστοποιησει την ανταμοιβή τους προς το σύστημα. Ωστόσο, λαμβάνοντας υπόψιν την απουσία γνώσης σχετικά με τα χαρακτηριστικά των ομάδων αλλα και ειδικότερα των χρηστών, γίνεται αντιληπτό ότι ο server δεν μπορεί να υπολογίσει τα πραγματικά rewards της κάθε ομάδας προς τις πιθανές συστάσεις (arms) του συστήματος. Αντ' αυτού, κρίνεται σκόπιμος ο υπολογισμός causal bounds για το reward που προσίδει στο σύστημα το κάθε arm. Μέσω των ορίων αυτών μπορεί να γίνει μεταφορά γνώσης μεταξύ των ομάδων και να προκύψει έτσι ένας νέος causal-constrained UCB αλγόριθμος, που θα επιτρέπει σε κάθε ομάδα χρηστών να μάθει την καλύτερη επιλογή που μπορεί να της προσφέρει το σύστημα αξιοποιώντας πληροφορίες από τις υπόλοιπες ομάδες. Ένα βασικό χαρακτηριστικό του αλγορίθμου μας είναι η ικανότητά του να αντιμετωπίζει σφάλματα κατά την ομαδοποίηση των χρηστών σε διαφορετικές ομάδες χρηστών, γεγονός πιθανό μιας και η ομάδα στην οποία ανήκει ο κάθε χρήστης θεωρείται πληροφορία άγνωστη στον server. Τέτοια σφάλματα μπορεί να οδηγήσουν σε user selection bias που μπορεί να επηρεάσει την διαδικασία εκμάθησης. Για να αντιμετωπίσουμε αυτήν την πρόκληση, εφαρμόζουμε μια ϵ-greedy μέθοδο, πουκειμένου τόσο να επιτραπεί η δίκαιη εξερεύνηση όων των arms του συστήματος όσο και να βελτιωθεί η ακρίβεια ομαδοποίησης κατά τη διάρκεια της εκμάθησης. Μέσω της διεξαγωγής αριθμητικών πειραμάτων αποδεικνύεται ότι ο προτεινόμενος αλγόριθμος ξεπερνά μεθόδους μάθησης που δεν λαμβάνουν υπόψη ούτε την ύπαρξη αιτιότητας (causality) στο σύστημα, αλλά ούτε και την ύπαρξη λάθους ομαδοποίησης των χρηστών του συστήματος.

Λέξεις-Κλειδιά: Εξατομικευμένο Σύστημα Συστάσεων, Ομαδοποίηση χρηστών, Συνεργατική μάθηση, Μεταφορά μάθησης, Αιτιότητα, Causal bounds, UCB

Abstract

In this paper we study a transfer learning problem for personalized recommendation systems to allow distinct user groups defined by distinct preferences to learn from each other without sharing private information. Specifically, we consider a server that has access only to user-click data points but not to any other personal user attributes. We model this system as a federated multi-armed bandit (MAB) problem with distinct bandit machines for each user group associated with distinct reward functions. The goal is to learn the true user types and rewards while maximizing the user clicks by adapting the arm selection strategy. Since, in the presence of unobserved user group preferences, the server cannot compute the true click-through rates (rewards) for each user group, we instead propose to compute causal bounds on the true click-through rates that by design contain the true values. Transferring these bounds to the different user groups, we obtain a new causal-bound-constrained UCB algorithm that allows each user group to learn their best arm by taking advantage of information from other groups. A key feature of our algorithm is its ability to deal with errors in the clustering of users to different user groups, which is possible since the true user group membership is assumed unknown to the server. Such errors can result in user selection bias and, therefore, bias in the learning process. To address this challenge, we implement an ϵ-greedy method to enable fair exploration and improve clustering accuracy during learning. We present numerical experiments that demonstrate that our proposed algorithm outperforms non-causal federated learning methods that do not account for clustering errors, especially for user types that constitute the minority of a data set.

Keywords: Recommendation System, Federated Learning, Personalization, Clustering, Causality, Transfer Learning, Causal Bounds, UCB.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους επιβλέποντες καθηγητές κ. Στάμου και κ. Ζαβλανό, καθώς και τους Διδακτορικούς φοιτητές του πανεπιστημίου Duke, Yi Shen και Yan Zhang, για την ευκαιρία που μου δόθηκε να διευρύνω τις γνώσεις μου μέσω της εκπόνησης αυτής της διπλωματικής και την καθοδήγησή τους κατά τη διάρκεια ολοκλήρωσής της.

Με αφορμή την ολοκλήρωση των σπουδών μου, θα ήθελα επίσης να ευχαριστήσω τους γονείς μου Αμαλία και Κώστα και την αδερφή μου Αλεξάνδρα για την σιωπηλή υπομονή και στήριξή τους. Θα ήθελα τέλος να ευχαριστήσω τους συμφοιτητές και φίλους μου Ανδρέα Ακαρέπη, Γιωργο Καλλίτση και Ηλιάνα Ξύγκου για την βοήθειά τους καθ' όλη τη διάρκεια των σπουδών μας και για τις αξέχαστες συνεργασίες μας.

Contents

List of Figures

List of Tables

Part I

Εκτεταμένη Περίληψη στα Ελληνικά

1 Ανάλυση Προβλήματος

Έστω ένα σύστημα συστάσεων που περιλαμβάνει M πιθανές προτάσεις, και του οποίου οι χρήστες κατανέμονται σε N διαφορετικούς τύπους χρηστών, σύμφωνα με μια σταθερή κατανομή P_U. Υποθέτουμε ότι ένας τύπος χρήστη u ορίζεται ως ο τύπος χρήστη που έχει εξερευνήσει την επιλογή u και έχει επίγνωση του γεγονότος ότι του αρέσει. Η ομάδα στην οποία ανήκει ο κάθε χρήστης, καθώς επίσης και οι προτιμήσεις του προς τις υπόλοιπες επιλογές του συστήματος μπορεί να επηρεαστούν από τα προσωπικά του στοιχεία και χαρακτηριστικά. Για παράδειγμα, στο [9] η συναισθηματική κατάσταση ενός χρήστη παρακολουθείται και χρησιμοποιείται προκειμένου το σύστημα να μπορεί να προτείνει ένα κατάλληλο είδος μουσικής. Τα μοντέλα που συνδέουν τα χαρακτηριστικά ενός χρήστη με το feedback του προς το σύστημα είναι ποικίλα. Για παράδειγμα, υπάρχουν περιπτώσεις στις οποίες θεωρείται ότι η προτίμηση ενός χρήστη προς μία πρόταση του συστήματος συνδέεται με τα χαρακτηριστικά του μέσω ενός γραμμικού μοντέλου [11], [8]. Προκειμένου να προστατεύσουμε το απόρρητο των χρηστών διατηρώντας κρυφά τα χαρακτηριστικά τους και επίσης να μην κάνουμε υποθέσεις σχετικά με το είδος του μοντέλου που συνδέει τα χαρακτηριστικά των χρηστών με τις προτιμήσεις τους (π.χ. γραμμικό μοντέλο), θεωρούμε τον τύπο χρήστη ως τη μόνη πληροφορία που μπορεί να χρησιμοποιηθεί κατά τη διάρκεια της μάθησης και τον αντιμετωπίζουμε ως το αίτιο που επηρεάζει τις προτιμήσεις των χρηστών προς τις προτάσεις του συστήματος.

Όπως σε κάθε σύστημα προτάσεων, ο στόχος είναι να μεγιστοποιηθεί η συνολική ανταμοιβή, η οποία σε αυτήν την περίπτωση ισοδυναμεί με το αν άρεσε σε έναν χρήστη μία επιλογή που του προτείνεται (με ανταμοιβή 1) ή όχι (με ανταμοιβή 0). Ωστόσο, δεδομένου ότι υποθέτουμε ότι οι χρήστες έχουν ήδη εξερευνήσει μία επιλογή που τους αρέσει, οι κύριοι στόχοι τους είναι να καθορίσουν τις προτιμήσεις τους τις υπόλοιπες προτάσεις του συστήματος και να αποφασίσουν ποια από αυτές θα μπορούσε να είναι η δεύτερη βέλτιστη επιλογή τους. Διατυπώνουμε το πρόβλημα ως ένα MAB που ορίζεται από την πλειάδα $\big(U, X, Y^U(X)\big)$, όπου η $U \in \{1, \dots, N\}$ είναι μια τυχαία μεταβλητή που μοντελοποιεί τον τύπο χρήστη, το $X \in \{1, \dots, M\}$ είναι μια τυχαία μεταβλητή που υποδεικνύει την επιλογή ενός από τα M arms και το $Y^u(x)$ είναι η συνάρτηση ανταμοιβής που σχετίζεται με τον βραχίονα $X = x$ δεδομένου του τύπου χρήστη $U = u$. Συγκεκριμένα, το $Y^u(x)$ αντιπροσωπεύει την πιθανότητα να αρέσει σε έναν χρήστη τύπου u η επιλογή x. Σε κάθε χρονικό βήμα, επιλέγεται ένας χρήστης τύπου u σύμφωνα με την κατανομή P_U. Με βάση τις πληροφορίες που παρέχει ο server για τον συγκεκριμένο χρήστη, ο τοπικός client επιλέγει μία από τις επιλογές που πρέπει να εξερευνηθούν από τον χρήστη και του την προτείνει. Η ανταμοιβή που προκύπτει ακολουθεί τη συνάρτηση προτιμήσεων του χρήστη Y^u.

Δεδομένου ότι οι χρήστες μπορούν να ταξινομηθούν σε ομάδες ανάλογα με τον τύπο τους, σχεδιάζουμε μία federated αρχιτεκτονική που μπορεί να επιτρέψει σε παρόμοιους χρήστες να εκπαιδεύσουν συνεργατικά το κοινό μοντέλο προτιμήσεών τους, αξιοποιώντας παράλληλα τις πληροφορίες που αποκτώνται από άλλους τύπους χρηστών. Σύμφωνα με αυτή τη διατύπωση, μπορεί να θεωρηθεί ότι για κάθε χρήστη υπάρχει ένας τοπικός client που του κάνει συστάσεις και ότι ολόκληρη η διαδικασία μάθησης εποπτεύεται από έναν κεντρικό server. Από την πλευρά του server, τα μόνα διαθέσιμα δεδομένα είναι data-points της μορφής user-arm-click, με τον τύπο του χρήστη να παραμένει ένας παράγοντας άγνωστος προς τον server. Ως αποτέλεσμα, από την πλευρά του server, οι μόνες πληροφορίες που μπορούν να υπολογιστούν με βεβαιότητα είναι causal bounds που περιορίζουν την ανταμοιβή που αντιστοιχεί στα arms του συστήματος [22]. Επομένως, ο κύριος στόχος είναι να σχεδιαστεί μια μέθοδος που βοηθά τον server να υπολογίσει εξατομικευμένα causal bounds για κάθε arm και για κάθε τύπο χρήστη και περαιτέρω να ενισχύσει τη διαδικασία μάθησης από την πλευρά των χρηστών.

2 Περιγραφή Αλγορίθμου

Σε αυτή την ενότητα προτείνουμε έναν αλγόριθμο βασισμένο στον UCB που λαμβάνει υπόψη το γεγονός ότι οι χρήστες ανήκουν σε διαφορετικούς τύπους χρηστών και χρησιμοποιεί αυτή την πληροφορία για να διευκολύνει τη διαδικασία μάθησης. Εξετάζουμε την προτεινόμενη μέθοδο τόσο από την πλευρά του χρήστη όσο και από την πλευρά του server. Αρχικά, αναλύουμε τον τρόπο με τον οποίο ο server χρησιμοποιεί τα δεδομένα της μορφής user-arm-click που αποστέλλονται από τους πελάτες, προκειμένου να ομαδοποιήσει τους χρήστες σε διαφορετικές ομάδες χρηστών. Στη συνέχεια, περιγράφουμε τη μέθοδο για τον υπολογισμό τόσο confidence όσο και των causal ορίων για κάθε ομάδα χρηστη, τα οποία θα λειτουργήσουν ως μέσο μεταφοράς μάθησης τόσο μεταξύ χρηστών του ίδιου τύπου όσο και διαφορετικών τύπων χρηστών αντίστοιχα. Τέλος, εξηγούμε τη διαδικασία δημιουργίας δεδομένων από την πλευρά των πελατών, η οποία επηρεάζεται κυρίως από τον αλγόριθμο causal-UCB 4 που εφαρμόζεται σε κάθε πελάτη.

2.1 Ανάλυση αλγορίθμου από την πλευρά του Server

Κατά τη διαδικασία μάθησης, ο server έχει πρόσβαση μόνο σε δεδομένα της μορφής (u, a, r) που συγκεντρώθηκαν από τους χρήστες, όπου a είναι το arm που εξετάστηκε και r η ανταμοιβή που δόθηκε (1 ή 0). Δεδομένου ότι δεν παρέχονται πληροφορίες σχετικά με τον τύπο του χρήστη, οι μόνες βέβαιες γνώσεις που μπορούν να υπολογιστούν στον server είναι τα causal bounds στην αναμενόμενη ανταμοιβή ενός arm [22]. Σε αυτήν την περίπτωση, έστω $p_{xy} = \Pr(X = x, Y = y)$ η πιθανότητα το arm x να έχει την ανταμοιβή y, η οποία μπορεί να υπολογιστεί χρησιμοποιώντας τα δεδομένα που αποστέλλονται στο server από τους πελάτες.

θεωρούμε $X \in \{1, \dots M\}$, $Y\{0, 1\}$, καθώς και ότι το p_{xy} δίνεται είτε ως εμπειρικά εκτιμώμενη τιμή είτε ως εκ των προτέρων γνώση. Τότε η αναμενόμενη ανταμοιβή ενός arm X χωρίς γνώση των συγκεκριμένων τύπων των χρηστών, $\mathbb{E}[Y|do(X)], το$ μπορεί να οριοθετηθεί από:

$$\Pr(X = x, Y = 1) \leq \mathbb{E}[Y|do(X = x)] \leq 1 - \Pr(X = x, Y = 0) \tag{1}$$

2.1.1 Ομαδοποίηση των χρηστών στον Server

Προκειμένου τα causal και upper confidence bounds που υπολογίζει ο server να αληθεύουν για κάθε χρήστη, είναι απαραίτητη η ομαδοποίηση των χρηστών και ο διαχωρισμός τους σε ομάδες παρόμοιων ενδιαφερόντων. Ο αλγόριθμος που χρησιμοποιείται για αυτή τη διαδικασία, 3, είναι ένα heuristic που χρησιμοποιεί τα δεδομένα που παρέχονται από τους χρήστες, για να τους ομαδοποιήσει σε συμπλέγματα σύμφωνα με τις προτιμήσεις τους. Συγκεκριμένα, ο server υπολογίζει ένα διάνυσμα x για κάθε χρήστη που αντιπροσωπεύει τη μέση ανταμοιβή του για κάθε arm. Στη συνέχεια, με τη χρήση του αλγόριθμου ομαδοποίησης KMeans, ο server μπορεί να ομαδοποιήσει τους χρήστες σε N ομάδες. Αυτή η ομαδοποίηση είναι αρκετή ώστε ο server να παρέχει στους χρήστες όλα τα απαραίτητα άνω όρια, τα οποία χρειάζονται για τον αλγόριθμο causal-UCB που θα εφαρμοστεί από την πλευρά των πελατών.

Επιπλέον, σε περίπτωση που ο τρόπος που ορίζονται οι τύποι των χρηστών, σε αυτήν τη διατύπωση προβλήματος, είναι μια πληροφορία γνωστή στον server, ο αλγόριθμος 3 παρέχει επίσης heuristic που καθορίζει ποια ομάδα, από αυτές που δημιουργούνται από το KMeans, αντιστοιχεί σε ποιον τύπο χρήστη. Η βασική ιδέα αυτού του αλγορίθμου βασίζεται στον ορισμό των τύπων χρήστη. Ως εκ τούτου, η επισήμανση των ομάδων επηρεάζεται από το ποιο arm έχει τη μεγαλύτερη ανταμοιβή στο κέντρο κάθε ομάδας, ενώ διασφαλίζεται ότι κανένας τύπος χρήστη δεν έχει εκχωρηθεί σε δύο διαφορετικές ομάδες.

Algorithm 1 Clustering and Labeling Algorithm

Require: For each user u a vector x_u with their average reward for each arm

1: Use KMeans to separate users into N groups based on their vector x_u.
2: For each group i created, compute the Center of the group c_i, which portrays the average rewards per arm of the users that were assigned at the mentioned group.
3: Assign each group to a user type by solving the following linear programming problem:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{N} -c_i y_i \\
\text{subject to} \quad & \sum_{k=1}^{M} c_i[k] y_i[k] = 1, \qquad i = 1, \ldots, N, \\
& \sum_{i=1}^{N} c_i[k] y_i[k] = 1, \qquad k = 1, \ldots, M, \\
& y_i[k] \in [0,1], \quad i = 1, \ldots, N \quad \text{and} \quad k = 1, \ldots M
\end{aligned}$$

4: Let l_i be the user-type user i according to the server
5: **for** $i = 1, \ldots, N$ **do**
6: $\quad l_i \leftarrow \arg\max_{k \in \{1, \ldots M\}} y_i[k]$, where $i \in \{1, \ldots N\}$
7: **end for**

2.1.2 Ανανέωση των Upper και Lower Confidence bounds των ανταμοιβών των arms

Μόλις ο server λάβει ένα data-point της μορφής (u, a, r) από έναν συγκεκριμένο χρήστη, το πρώτο βήμα είναι να ανανεώσει τα άνω και κάτω confidence όρια του arm a στον τύπο χρήστη που ανήκει ο u, σύμφωνα με τον server.

Έστω ένα data-point (u, a, r) που αποστέλλεται από τον χρήστη u, ο οποίος σύμφωνα με τον server ανήκει στον τύπο χρήστη k. Στη συνέχεια, εάν t_k είναι ο αριθμός των συνολικών δεδομένων που συλλέγονται για τον τύπο χρήστη k, $N_k[a]$ ο αριθμός των δεδομένων που συλλέχθηκαν για τον τύπο χρήστη k σχετικά με το αρμ a και $r_k[a]$ η συνολική ανταμοιβή που έχει συλλεχθεί μέχρι στιγμής για το arm a στον k τύπο χρήστη, το ucb και το lcb του $Y^k[a]$ μπορούν να ενημερωθούν σύμφωνα με τους ακόλουθους τύπους:

$$\text{UCB}(Y^k(a)) \leftarrow \frac{r_k[a]}{N_k[a]} + \sqrt{1.5 \frac{\log t_k}{N_k[a]}}, \tag{2}$$

$$\text{LCB}(Y^k(a)) \leftarrow \frac{r_k[a]}{N_k[a]} - \sqrt{1.5 \frac{\log t_k}{N_k[a]}}. \tag{3}$$

Παρά το γεγονός ότι υπολογίζονται τόσο τα άνω όσο και τα κάτω confidence bounds, οι τοπικοί πελάτες χρειάζονται μόνο το άνω όριο όταν πρόκειται να καθορίσουν ποιο arm θα πρέπει να προτείνουν στους χρήστες σε κάθε χρονικό βήμα. Ωστόσο, ο υπολογισμός του κατώτερου ορίου είναι επίσης απαραίτητος, καθώς αποτελεί έναν επιπλέον περιορισμό που θα χρησιμοποιήσει ο server κατά τον υπολογισμό των causal bounds. Κάθε φορά που αλλάζει το upper confidence bound της ανταμοιβής του τύπου k για ένα συγκεκριμένο arm, η ενημερωμένη έκδοσή του θα αποστέλλεται σε όλους τους χρήστες που σύμφωνα με τον server ανήκουν στον εξεταζόμενο τύπο χρήστη, k.

2.1.3 Υπολογισμός των Causal bounds για την αναμενόμενη ανταμοιβή των arms

Εάν λάβουμε υπόψη την ύπαρξη διαφορετικών τύπων χρήστη στο σύστημα που υπακούν στην κατανομή P_U, τότε η αναμενόμενη ανταμοιβή $\mathbb{E}$[Y—do(X)] μπορεί να επεκταθεί ως:

$$\mathbb{E}[Y|do(X)] = \sum_{i=1}^{N} P_U^i \mathbb{E}[Y^k|do(X)]. \tag{4}$$

Μόλις βρεθούν ή ενημερωθούν τα confidence bounds του $Y^k(a)$, είναι δυνατό να αποκτηθούν causal bounds για $\mathbb{E}[Y^i|do(X=a)]$, όπου i $\sigma\varepsilon\{1,\cdots,N\}\backslash\{k\}$, χρησιμοποιώντας τα causal bounds του $\mathbb{E}[Y|do(X=a)]$ στο (6) και την ανάλυση στο (9).

Έστω UCausal(a) και LCausal(a) τα άνω και κάτω causal bounds του $\mathbb{E}[Y|do(X=a)]$ όπως υπολογίζεται στο (6). Έστω P_U η κατανομή των τύπων χρήστη που έχει βρει ο server μέσω της ομαδοποίησης και UCB($Y^k(a)$), LCB($Y^k(a)$) τα ενημερωμένα confidence bounds για την ανταμοιβη του a σε τύπο χρήστη k. Στη συνέχεια, τα ανώτερα causal όρια για την ανταμοιβή του a σε τύπους χρήστη διαφορετικών από των k μπορούν να υπολογιστούν επιλύοντας το ακόλουθο πρόβλημα βελτιστοποίησης.

$$\begin{aligned}
\text{minimize} \quad & -Y^i(a) \\
\text{subject to} \quad & \sum_{j=1}^{N} P(U=j)\cdot Y^j(a) \le \text{UCausal}(a), \\
& \sum_{j=1}^{N} P(U=j)\cdot Y^j(a) \ge \text{LCausal}(a), \\
& Y^j(a) \in \left[\text{LCB}(Y^j(a)), \text{UCB}(Y^j(a))\right], \quad j=1,\ldots,N
\end{aligned}$$

Στο παραπάνω πρόβλημα βελτιστοποίησης, το $Y^i(a)$ αντιπροσωπεύει το άνω causal όριο της ανταμοιβής του arm a για τον i τύπο χρήστη. Ελαχιστοποιώντας την τιμή του $-Y^i(a)$, καταφέρνουμε να λάβουμε τη μέγιστη δυνατή τιμή για αυτό το άνω όριο. Προκειμένου να ληφθεί ένα άνω causal όριο για όλους τους τύπους χρήστη, είναι απαραίτητο να λυθεί αυτό το πρόβλημα βελτιστοποίησης N φορές για $i \in\in \{1,\cdots,N\}\backslash\{k\}$. Εάν δοθεί οποιαδήποτε προηγούμενη γνώση σχετικά με τους τύπους χρήστη και τις προτιμήσεις τους (π.χ. $\frac{Y^1(a)}{Y^2(a)} \in [0,3,0,6]$), μπορεί να χρησιμοποιηθεί ως ένας επιπλέον περιορισμός στο παραπάνω πρόβλημα βελτιστοποίησης για να βοηθήσει τα causal όρια να γίνουν πιο αυστηρά.

2.2 Ανάλυση αλγορίθμου από την πλευρά των Clients

Από την πλευρά των πελατών, θεωρείται ότι κάθε χρήστης γνωρίζει τον τύπο του, που σημαίνει ότι γνωρίζει ποια επιλογή του συστήματος του αρέσει ήδη και ποιες είναι οι υπόλοιπες που πρέπει να εξερευνήσει περαιτέρω. Ως αποτέλεσμα, κατά τη διάρκεια της μάθησης, σε κάθε χρήστη οι επιλογές που θα του παρουσιάζονται θα προέρχονται από ένα συγκεκριμένο pool of arms, ανάλογο του τύπου του χρήστη, και σε κάθε χρονικό βήμα το arm που θα επιλέγεται θα είναι αυτό με τις καλύτερες προοπτικές ανταμοιβής (οι προοπτικές ανταμοιβής καθορίζονται με βάση τα άνω όρια των arms). Τα data-points που δημιουργούνται από αυτή τη διαδικασία θα χρησιμοποιηθούν από τον server στη διαδικασία ομαδοποίησης των χρηστών. Προκειμένου να δοθεί η δυνατότητα στον server να ομαδοποιήσει σωστά τους χρήστες και να βελτιώσει την ακρίβεια ομαδοποίησης, είναι απαραίτητο στην αρχή του αλγορίθμου να πραγματοποιηθεί μια δίκαιη εξερεύνηση των επιλογών του συστήματος. Για να αντιμετωπίσουμε αυτήν την πρόκληση, εφαρμόζουμε μια μέθοδο ϵ-greedy, στην οποία το ϵ είναι μια παράμετρος συντονισμού που προσαρμόζει την εξερεύνηση του αλγορίθμου για κάθε χρήστη κατά τα πρώτα βήματα της προσομοίωσης.

Algorithm 2 Causal bound constrained ϵ-greedy-UCB

Require: List of Users U, Pool of arms X, UCB bounds on the rewards per arm for each user type $Y^U(X)$, causal bounds on $Y^U(X)$ (both bounds are provided by the server), number of time steps T

$\epsilon = 0.1$
for $t = 1, \ldots, T$ **do**
  Choose randomly one user u from U.
  Let k be the user's true type.
  $a \leftarrow \arg\max_x \min\{\text{UCB}(x), \text{UCausal}(x)\}$, where $x \in X \backslash k$
  $flag = \text{random.choice}([1, 0], [\epsilon, 1 - \epsilon])$
  if $flag == 1$ **then**
    $a \leftarrow \text{random.choice}([X])$
  end if
  $r \leftarrow \text{random.choice}([1, 0], [1 - Y^k(a), Y^k(a)])$, where r the reward provided
  Send the data point (u, a, r) to the server.
end for

Παρατηρήθηκε ότι χωρίς την ενσωμάτωση της μεθόδου ϵ-greedy στον αλγόριθμο, η ακρίβεια ομαδοποίησης από την πλευρά του server, ακόμη και σε περιπτώσεις όπου οι τύποι χρήστη διέφεραν μεταξύ τους, δεν θα μπορούσε να βελτιωθεί αρκετά κατά τη διαδικασία μάθησης. Μόλις τα causal bounds συγκλίνουν, κυριαρχούν έναντι των upper confidence bounds και αναγκάζουν τους τοπικούς clients να διαλέγουν και να προτείνουν πάντα το ίδιο arm. Όταν τα causal όρια ήταν σωστά, αυτός η επιλογή που θα γινόταν θα ήταν η βέλτιστη, διαφορετικά όχι. Και στις δύο περιπτώσεις, ωστόσο, οι χρήστες δεν θα είχαν την ευκαιρία να εξερευνήσουν τις υπόλοιπες επιλογές του συστήματος και έτσι ο server δεν θα είχε μια πιο ολοκληρωμένη εικόνα όλων των προτιμήσεών τους. Ως αποτέλεσμα, δεδομένου του ότι η ομαδοποίηση βασίζεται στην εκτίμηση του server για τις προτιμήσεις των χρηστών, η ακρίβειά της δεν θα μπορούσε να είναι ικανοποιητική.

Με τη μέθοδο ϵ-greedy, οι τοπικοί clients είχαν τη δυνατότητα να εξερευνήσουν όλες τις επιλογές του συστήματος και να αποκτήσουν μια καλύτερη ιδέα για τις προτιμήσεις των χρηστών. Με αυτόν τον τρόπο, ο server μπόρεσε να κατανοήσει τα ενδιαφέροντά τους και έτσι να καταφέρει να ομαδοποιήσει τους χρήστες με μεγαλύτερη ακρίβεια, τουλάχιστον στις περιπτώσεις που οι τύποι χρηστών ήταν διαφορετικοί μεταξύ τους. Τόσο η παράμετρος ϵ όσο και ο χρόνος κατά τον οποίο η μέθοδος ϵ-greedy θα σταματήσει να είναι ενεργή, είναι tuning parameters που μπορούν να οριστούν σύμφωνα με την εξεταζόμενη προσομοίωση και το πρόβλημα.

3 Πειράματα και Αποτελέσματα

Έστω ένα σύστημα προτάσεων του οποίου οι χρήστες μπορούν να ομαδοποιηθούν σε 3 suer-types ($N = 3$) και μπορούν να επιλέξουν άρθρα από μια ομάδα 3 arms ($M = 3$). Κάθε arm αντιπροσωπεύει μια διαφορετική κατηγορία άρθρων και κάθε τύπος χρήστη k ορίζεται ως η ομάδα χρηστών που έχουν εξερευνήσει το arm k και γνωρίζουν ότι είναι η βέλτιστη επιλογή τους. Ο επόμενος στόχος των χρηστών είναι να εξερευνήσουν τα υπόλοιπα arms και να μάθουν ποια είναι η δεύτερη καλύτερη επιλογή τους.

Θεωρούμε ότι υπάρχουν 100 χρήστες συνδεδεμένοι στο σύστημα και ότι χωρίζονται στις 3 ομάδες σύμφωνα με την κατανομή $P_{U_1} = 0,2$, $P_{U_2} = 0,3$ και $P_{U_3} = 0,5$. Πριν ξεκινήσει η διαδικασία εκμάθησης, το σύστημα συλλέγει έναν αριθμό αρχικών data-points ανά χρήστη, προκειμένου ο server να αρχικοποιήσει την ομαδοποίηση. Αυτά τα δεδομένα χρησιμοποιούνται επίσης για τον προσδιορισμό των αρχικών τιμών των ορίων ucb ανά τύπο χρήστη. Συγκεκριμένα, στα πειράματα που πραγματοποιήθηκαν, κάθε χρήστης παρέχει πέντε αρχικά δεδομένα που συλλέγονται με βάση την τοπική πολιτική που προτείνει το καλύτερο arm για κάθε χρήστη με πιθανότητα 0,6 και ένα από τα άλλα δύο arms με πιθανότητα 0,2 το καθένα.

Μετά την προετοιμασία τόσο της ομαδοποίησης στον server όσο και των άνω και κάτω ορίων εμπιστοσύνης, ο αλγόριθμος εκτελείται για 200000 χρονικά βήματα. Σε κάθε χρονικό βήμα, ο T, ένας από τους 100 χρήστες επιλέγεται τυχαία και του παρουσιάζεται ένας βραχίονας σύμφωνα με το μοντέλο που του παρείχε ο διακομιστής. Έστω ότι ο χρήστης που εξετάστηκε ανήκε στον τύπο χρήστη k και ο βραχίονας που του δόθηκε ήταν βραχίονας a. Στη συνέχεια, το regret του τύπου χρήστη k, για τον οποίο η καλύτερη ανταμοιβή είναι μ^k, στο χρονικό βήμα T υπολογίζεται σύμφωνα με την ακόλουθη εξίσωση:

$$\text{regret}_T^k = T\mu^k - \sum_{t=1}^{T} Y^k(a) \tag{5}$$

Για να είναι γενικά έγκυρα τα αποτελέσματα, η κύρια προσομοίωση πραγματοποιείται 25 φορές και το τελικό regret που παρουσιάζεται αντιπροσωπεύει το μέσο regret όλων των δοκιμών.

Τα αποτελέσματα των πειραμάτων συγκρίνονται με αυτά δύο benchmarks. Το πρώτο εξακολουθεί να προσπαθεί να ομαδοποιήσει τους χρήστες στην πλευρά του διακομιστή και να εκπαιδεύσει το μοντέλο τους συλλογικά, αλλά αγνοεί την ύπαρξη άγνωστων πληροφοριών και unobserved context. Ως αποτέλεσμα, δεν υπολογίζονται ούτε χρησιμοποιούνται causal bounds κατά τη διάρκεια της μαθησιακής διαδικασίας και έτσι καμία πληροφορία που λαμβάνεται από μια ομάδα δεν μεταφέρεται σε άλλη. Το δεύτερο benchmark δεν λαμβάνει υπόψη ότι οι χρήστες μπορεί να έχουν διαφορετικά μοντέλα και συνεπώς προτιμήσεις και χρησιμοποιεί ομοσπονδιακή μάθηση για να εκπαιδεύσει ένα κοινό μοντέλο για όλους τους χρήστες χωρίς να επιχειρήσει ομαδοποίηση ή αιτιακή μεταφορά.

Σε αυτήν την ενότητα, υπάρχουν δύο κύρια πειράματα που διεξάγονται. Η πρώτη εξετάζει τους τύπους χρηστών που είναι αρκετά διαφορετικοί μεταξύ τους και σε κάθε τύπο χρήστη η δεύτερη βέλτιστη επιλογή διαφέρει πολύ από την τελευταία. Στο δεύτερο πείραμα, οι τύποι χρηστών έχουν πολλές ομοιότητες μεταξύ τους, παρόλο που τα πραγματικά τους μοντέλα είναι θεμελιωδώς διαφορετικά.

3.1 Τύποι χρήστη που διαφέρουν μεταξύ τους

Στην αρχή του πειράματος, είναι σημαντικό να καθοριστούν τα αληθινά μοντέλα που επηρεάζουν τις ανταμοιβές των χρηστών προς τους βραχίονες του συστήματος. Αρχικά, κάθε ένας από τους 100 χρήστες του συστήματος εκχωρείται σε έναν από τους 3 τύπους χρηστών, σύμφωνα με την κατανομή $P_{U_1} = 0,2$, $P_{U_2} = 0,3$ και $P_{U_3} = 0,5$. Στη συνέχεια, οι προτιμήσεις των τύπων χρήστη προς τα arms αντιπροσωπεύονται από τις πιθανότητες που εμφανίζονται στον πίνακα **??**. Επομένως, το αληθινό μοντέλο που ορίζεται για κάθε τύπο χρήστη, είναι στην πραγματικότητα η πιθανότητα ένας χρήστης τύπου k να κάνει κλικ σε έναν βραχίονα a και έτσι να του παρέχει μια ανταμοιβή ίση με 1.

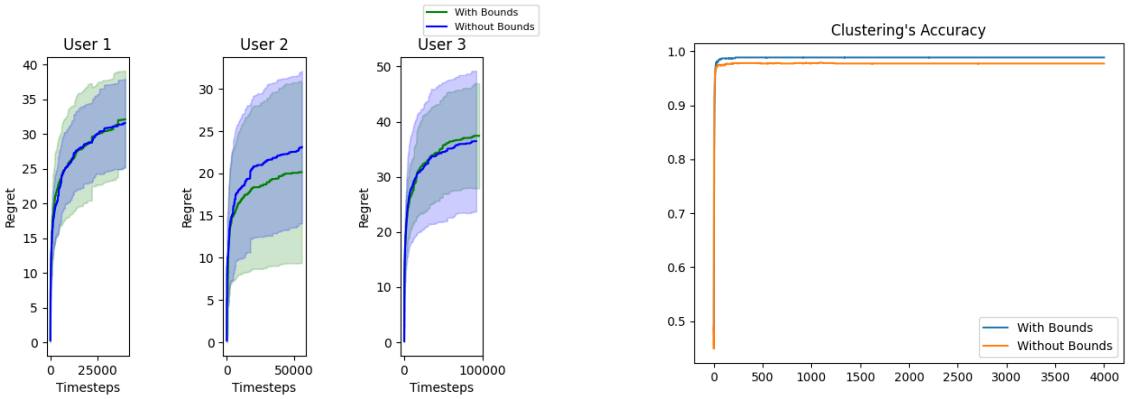
Στον αλγόριθμο που περιγράφεται παραπάνω, είναι πιθανό ο διακομιστής να έχει προηγούμενες γνώσεις για τα μοντέλα κάθε τύπου χρήστη και, πιο συγκεκριμένα, για τη σχέση μεταξύ των μοντέλων. Αυτή η γνώση θα μπορούσε να θεωρηθεί ως a priori πληροφορίες που παρέχονται από στατιστικές ή άλλες έρευνες. Τέτοιες πληροφορίες θα μπορούσαν να χρησιμοποιηθούν στον υπολογισμό των causal bounds ως επιπλέον περιορισμός στο πρόβλημα βελτιστοποίησης που περιγράφεται στην ενότητα **3.2.3**. Σε αυτή την ενότητα, εξετάζουμε επίσης εάν αυτή η γνώση, εάν είναι σωστή, μπορεί να επιταχύνει τελικά τη σύγκλιση των causal bounds και επομένως των μοντέλων που μαθαίνονται στους πελάτες, ή εάν τα σφάλματα θα επηρεάσουν την ακρίβεια του αλγορίθμου.

Table 1: Probability table that describes the preferences per user type.

	$User\_type1$	$User\_type2$	$User\_type3$
$Pr(X = arm_1, Y = 1)$	1	0.2	0.8
$Pr(X = arm_2, Y = 1)$	0.8	1	0.4
$Pr(X = arm_3, Y = 1)$	0.3	0.9	1

3.1.1 Χωρίς επιπλέον περιορισμούς

Αρχικά, στη διατύπωση που περιγράφηκε παραπάνω, ο αλγόριθμος υλοποιείται χωρίς τη χρήση επιπλέον πληροφοριών ή περιορισμών στα μοντέλα των χρηστών. Ο κύριος αλγόριθμος του paper συγκρίνεται με αυτόν ενός απλού UCB που υλοποιεί επίσης ομαδοποίηση στον διακομιστή, αλλά δεν υπολογίζει ούτε χρησιμοποιεί κανένα αιτιολογικό όριο ενώ προτείνει όπλα στους πελάτες. Η απόδοσή τους αξιολογείται με βάση το διάγραμμα regret, στο οποίο το regret υπολογίζεται όπως φαίνεται στην εξίσωση 10, και την ακρίβεια της ομαδοποίησης από την πλευρά του διακομιστή.



(a) With and Without causal bounds

(b) Clustering Accuracy

Figure 1: Comparison of the algorithms with and without causal bounds, while not using any constraints.

Από αυτή την προσομοίωση, μπορεί να παρατηρηθεί ότι παρόλο που η χρήση causal bounds μπορεί να επιταχύνει τη διαδικασία εκμάθησης για ορισμένους χρήστες (σε αυτήν την περίπτωση για χρήστες που ανήκουν στον τύπο 2), γενικά δεν θα επηρεάσει δραστικά την απόδοση του συστήματος. Είναι γνωστό ότι κατά τον υπολογισμό των causal bounds, τα όρια εμπιστοσύνης χρησιμοποιούνται ως επιπλέον περιορισμός στο πρόβλημα βελτιστοποίησης. Επομένως, η σύγκλιση των αιτιακών ορίων εξαρτάται από τη σύγκλιση των ορίων εμπιστοσύνης και ως αποτέλεσμα, όταν δεν υπάρχουν άλλες πληροφορίες που μπορούν να χρησιμοποιηθούν στο πρόβλημα βελτιστοποίησης, τα αιτιακά όρια δεν μπορούν να βελτιώσουν δραστικά τα τελικά αποτελέσματα του συστήματος.

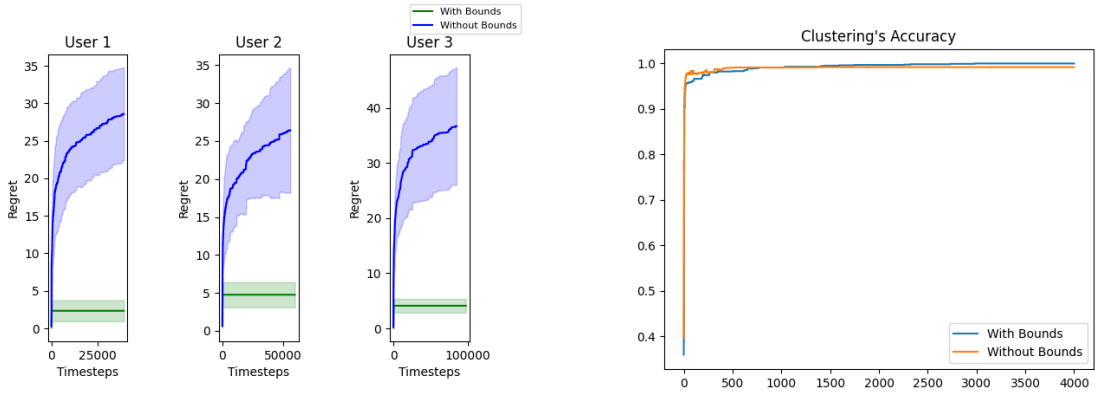
3.1.2 Σωστοί περιορισμοί

Ενώ τα causal bounds διευκολύνουν ορισμένους χρήστες να μαθαίνουν το μοντέλο τους πιο γρήγορα, γενικά δεν κάνουν μεγάλη διαφορά στην απόδοση του αλγορίθμου, καθώς η βελτίωσή τους είναι αλληλένδετη με τη βελτίωση των ορίων εμπιστοσύνης. Ως εκ τούτου, είναι προφανές ότι οποιαδήποτε σωστή προηγούμενη πληροφορία θα μπορούσε να βοηθήσει τα causal bounds να γίνουν πιο αυστηρά και επομένως πιο χρήσιμα πιο γρήγορα από τα όρια εμπιστοσύνης.

Για τον λόγο που εξηγήθηκε παραπάνω, σε αυτή την ενότητα εξετάζουμε την περίπτωση όπου ο διακομιστής έχει γνώση σχετικά με τη σχέση μεταξύ των μοντέλων των χρηστών. Συγκεκριμένα, κάθε φορά που ο διακομιστής επιλύει το πρόβλημα βελτιστοποίησης που περιγράφεται στην ενότητα **3.2.3**, χρησιμοποιεί επίσης τους ακόλουθους περιορισμούς:

- $0.05 \leq \frac{Y^2(1)}{Y^3(1)} \leq 0.6$
- $0.3 \leq \frac{Y^3(2)}{Y^1(2)} \leq 0.7$
- $0.1 \leq \frac{Y^1(3)}{Y^2(3)} \leq 0.6$

Like before, the paper's algorithm is compared to the one that neither computes nor uses causal bounds.



(a) With and Without causal bounds (b) Clustering Accuracy

Figure 2: Comparison of the algorithms with and without causal bounds, while using correct constraints.

Στην περίπτωση που προηγούμενες πληροφορίες για τις προτιμήσεις των χρηστών δίνονται στον διακομιστή και χρησιμοποιούνται ως επιπλέον περιορισμός στο πρόβλημα βελτιστοποίησης, είναι προφανές ότι η χρήση causal bounds επιτρέπει στο σύστημα να συγκλίνει σχεδόν αμέσως στην πραγματική του κατάσταση. Έχει παρατηρηθεί ότι όταν αυτοί οι επιπλέον περιορισμοί είναι ενεργοί, τα causal bounds συγκλίνουν στην τελική τους τιμή μετά από λίγα μόνο σημεία δεδομένων. Όταν οι περιορισμοί που παρέχονται είναι αληθείς, οι τελικές τιμές των ορίων

είναι επίσης σωστές και προσεγγίζουν τις προτιμήσεις των χρηστών, επιτρέποντας έτσι στους τοπικούς συστάσεις να κάνουν ακριβείς προτάσεις στους χρήστες.

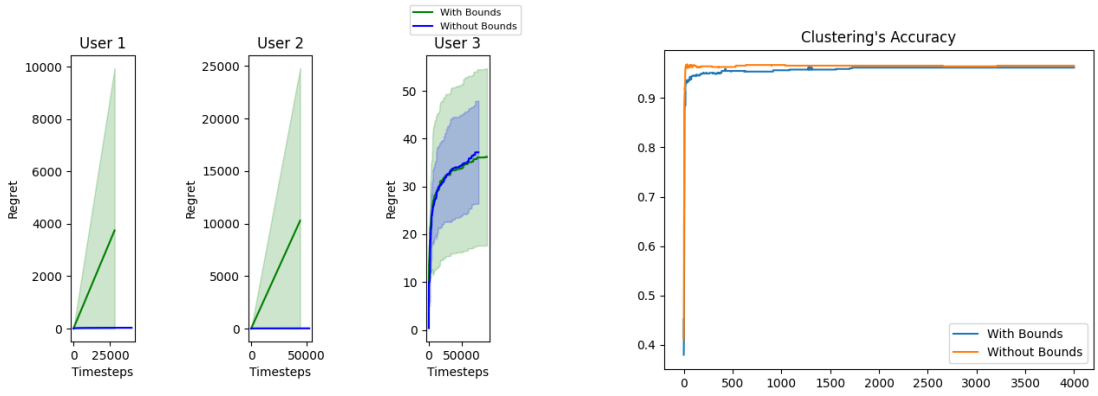
3.1.3 Λάθος περιορισμοί που επηρεάζουν την απόδοση του συστήματος

Παρόλο που υπάρχουν περιπτώσεις που οι σωστοί περιορισμοί μπορούν ενδεχομένως να βοηθήσουν τους πελάτες να συγκλίνουν στο πραγματικό τους μοντέλο πιο γρήγορα, η ευρωστία του αλγορίθμου έναντι εσφαλμένων περιορισμών δεν είναι εγγυημένη. Η χρήση λανθασμένης προηγούμενης γνώσης θα επηρεάσει πιθανώς τις τιμές των αιτιακών ορίων και αναπόφευκτα θα επιδεινώσει την απόδοση των αλγορίθμων.

Σε αυτήν την ενότητα, εξετάζουμε την περίπτωση κατά την οποία η προηγούμενη γνώση που μεταφέρθηκε στον διακομιστή είναι εσφαλμένη, και στην πραγματικότητα για τους βραχίονες 2 και 3, αντιπροσωπεύει την αντίθετη σχέση από την αληθινή, όπως φαίνεται στους ακόλουθους περιορισμούς:

- $0.05 \leq \frac{Y^2(1)}{Y^3(1)} \leq 0.6$
- $0.3 \leq \frac{Y^1(2)}{Y^3(2)} \leq 0.7$
- $0.1 \leq \frac{Y^2(3)}{Y^1(3)} \leq 0.6$

The comparison between the algorithm with the wrong constraints and the one that does not use any causal bounds is shown in figure 17



(a) With and Without causal bounds

(b) Clustering Accuracy

Figure 3: Comparison of the algorithms with and without causal bounds, while using incorrect constraints.

Σε αυτή την προσομοίωση, είναι προφανές ότι εσφαλμένοι περιορισμοί ενδέχεται να διαταράξουν τα τελικά αποτελέσματα του αλγορίθμου. Όταν δεν διασφαλίζεται η αληθινή σχέση μεταξύ των προτιμήσεων των χρηστών, είναι δυνατό για τους local clients να κάνουν προτάσεις στους χρήστες που δεν είναι κατάλληλες για τον τύπο τους, επιτρέποντας έτσι στο regret του αλγορίθμου να συνεχίσει να αυξάνεται κατά τη διάρκεια όλης της μαθησιακής διαδικασίας. Η απόδοση του συστήματος δεν αλλάζει μόνο για τους τύπους χρηστών που συνδέονται με λάθος πληροφορίες, σε αυτήν την περίπτωση ο χρήστης πληκτρολογεί 1 και 2, αλλά και για τους υπόλοιπους τύπους χρήστη 3 για αυτήν την προσομοίωση. Δεδομένου ότι το causal όριο για την προτίμηση ενός τύπου χρήστη k για βραχίονα a ενημερώνεται κάθε φορά που ένας άλλος τύπος χρήστη παρέχει πληροφορίες για το συγκεκριμένο σκέλος, εάν η αλληλεπίδραση τουλάχιστον ενός από τους τύπους χρήστη αλλάξει, τότε το σύνολο επηρεάζεται το σύστημα.

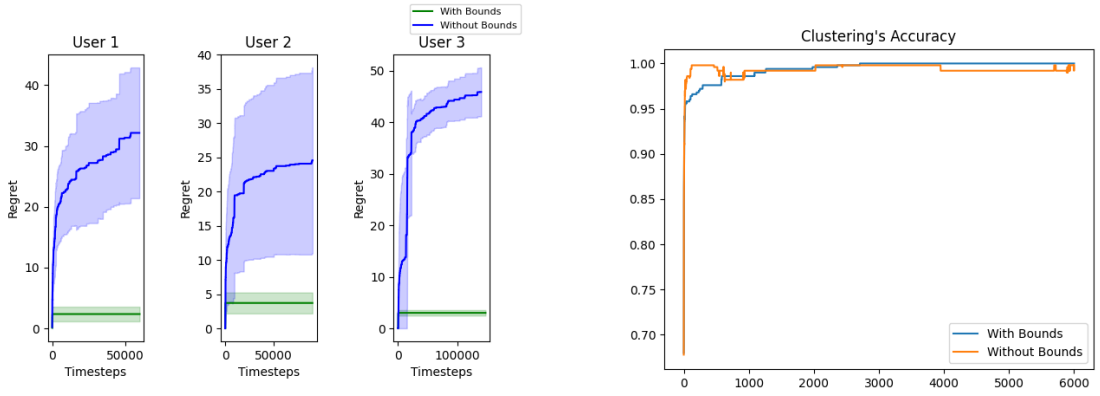
3.1.4 Λάθος περιορισμοί που δεν επηρεάζουν την απόδοση του συστήματος

Ο κύριος στόχος του αλγορίθμου είναι να προσδιορίσει ποιο από τα δύο υπόλοιπα arms, που δεν έχουν διερευνηθεί ακόμη από τους χρήστες, τους αρέσει περισσότερο. Στη συνέχεια δίνεται το ελάχιστη regret εάν ο τοπικός πελάτης προτείνει το optimal άρθρο στους χρήστες. Ως εκ τούτου, είναι πιθανό ότι ακόμη και αν οι προηγούμενες γνώσεις που δίνονται στον διακομιστή είναι εσφαλμένες, και ως αποτέλεσμα τα όρια των προτιμήσεων των χρηστών προς τον βραχίονα είναι λανθασμένα, ο διακομιστής εξακολουθεί να καταφέρει να μάθει τον βέλτιστο βραχίονα των χρηστών. Ανεξάρτητα από το ποια είναι η εκτίμηση του διακομιστή (σωστή ή όχι) για τις πραγματικές ανταμοιβές των τύπων χρήστη για τους βραχίονες, εφόσον εξακολουθεί να μπορεί να μάθει ποιος βραχίονας είναι καλύτερος από τον άλλο, το σύστημα θα επιτύχει τον κύριο σκοπό του.

Προκειμένου να αποδείξουμε ότι μπορεί να υπάρχει μια τέτοια περίπτωση, προσομοιώσαμε τη διατύπωση που περιγράφηκε παραπάνω και για το πρόβλημα βελτιστοποίησης που επιλύθηκε από τον διακομιστή, δώσαμε τους ακόλουθους λανθασμένους περιορισμούς:

- $0.4 \leq \frac{Y^2(1)}{Y^3(1)} \leq 0.6$
- $0.6 \leq \frac{Y^3(2)}{Y^1(2)} \leq 0.7$
- $0.4 \leq \frac{Y^1(3)}{Y^2(3)} \leq 0.6$

Όπως και πριν, ο αλγόριθμος της εργασίας συγκρίνεται με αυτόν που δεν λαμβάνει υπόψη την ύπαρξη αιτιότητας στο πρόβλημα, και επομένως δεν χρησιμοποιεί κανένα causal bound.



(a) With and Without causal bounds

(b) Clustering Accuracy

Figure 4: Comparison of the algorithms with and without causal bounds, while using incorrect constraints.

Από το σχήμα 18, μπορεί να παρατηρηθεί ότι παρόλο που ο διακομιστής είχε λανθασμένο περιορισμό σχετικά με τις προτιμήσεις των χρηστών, η χρήση αιτιακών ορίων εξακολουθεί να είναι αρκετά χρήσιμη για τη σύγκλιση του συστήματος. Αν και η τιμή της ανταμοιβής που θα περίμενε ο διακομιστής ($E[Y^u|do(X)]$) δεν είναι σωστή, ο διακομιστής εξακολουθεί να καταφέρνει να μάθει ποιες είναι οι καλύτερες προτάσεις για κάθε χρήστη.

3.1.5 Σύγκριση με έναν αλγόριθμο UCB που δεν κάνει ομαδοποίηση

Μέχρι στιγμής, σε όλα τα παραπάνω πειράματα, θεωρήθηκε ότι ο διακομιστής γνώριζε την ύπαρξη τύπων χρήστη, και ακόμη περισσότερο, τον πραγματικό αριθμό των ομάδων του συστήματος. Επειδή η σημασία της ομαδοποίησης στον διακομιστή μπορεί να αμφισβητηθεί, θεωρήσαμε σκόπιμο να συγκρίνουμε τον προτεινόμενο αλγόριθμο με τα αποτελέσματα που παρέχονται από ένα απλό UCB που αγνοεί την ύπαρξη τύπων χρήστη στο σύστημα και δεν επιχειρεί καμία ομαδοποίηση. Σε αυτόν τον αλγόριθμο, ο διακομιστής παρέχει ένα σύνολο ανώτερων ορίων εμπιστοσύνης που είναι το ίδιο για όλους τους χρήστες, ανεξάρτητα από την ομάδα χρηστών που ανήκουν. Η σύγκριση των δύο αλγορίθμων φαίνεται στο σχήμα 19.

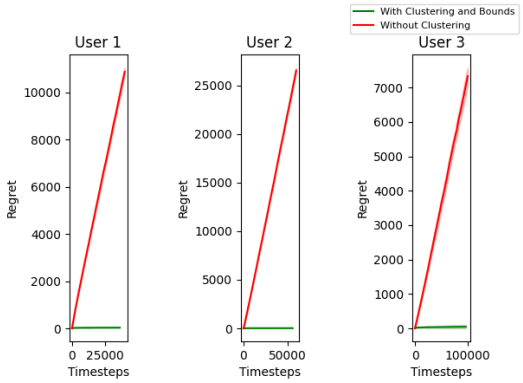


Figure 5: With and Without Clustering at the Server

Είναι αντιληπτό ότι όταν δεν πραγματοποιείται ομαδοποίηση, καθώς οι προτιμήσεις των χρηστών διαφέρουν, ο συνδυασμός των δεδομένων τους για την εκμάθηση ενός κοινού μοντέλου δεν μπορεί να οδηγήσει στα επιθυμητά αποτελέσματα. Σε αυτήν την περίπτωση, για παράδειγμα, κανένας από τους τοπικούς πράκτορες δεν καταφέρνει να μάθει το πραγματικό μοντέλο των τριών τύπων χρηστών. Παρόλο που ο διακομιστής αγνοεί την ύπαρξη ομάδων χρηστών, οι τοπικοί πράκτορες συνεχίζουν να κάνουν μεροληπτικές προτάσεις στους χρήστες ανάλογα με τον τύπο τους. Αυτή η μεροληπτική και περιορισμένη συλλογή σημείων δεδομένων, σε συνδυασμό με το γεγονός ότι ο διακομιστής δεν αναγνωρίζει ότι το σύνολο δεδομένων προέρχεται από διαφορετικές διανομές, αναγκάζει το σύστημα να συγκλίνει σε λάθος τελική κατάσταση.

3.2 Παρόμοιοι τύποι χρηστών

Στα προηγούμενα πειράματα, οι τύποι χρηστών που ορίστηκαν ήταν αρκετά διαφορετικοί και οι προτιμήσεις τους διέφεραν μεταξύ τους. Ωστόσο, μια άλλη ενδιαφέρουσα περίπτωση που μπορεί να εξεταστεί είναι όταν οι χρήστες διαφορετικών τύπων έχουν πράγματι παρόμοια γεύση. Όπως και πριν, στην αρχή του πειράματος είναι απαραίτητο να οριστούν τα αληθινά μοντέλα που επηρεάζουν τις ανταμοιβές των χρηστών προς τα χέρια του συστήματος. Αρχικά, κάθε ένας από τους 50 χρήστες του συστήματος εκχωρείται σε έναν από τους 3 τύπους χρηστών, σύμφωνα με την ίδια κατανομή (που σημαίνει $P_{U_1} = 0,2$, $P_{U_2} = 0,3$ και $P_{U_3} = 0,5$). Στη συνέχεια, οι προτιμήσεις των τύπων χρήστη προς τους βραχίονες αντιπροσωπεύονται από τις πιθανότητες που εμφανίζονται στον πίνακα 4. Όπως σε όλα τα προηγούμενα πειράματα, κάθε φορά που ένας χρήστης αρέσει σε ένα άρθρο και κάνει κλικ σε αυτό, του παρέχει μια ανταμοιβή ίση με 1

Table 2: Probability table that describes the preferences per user type.

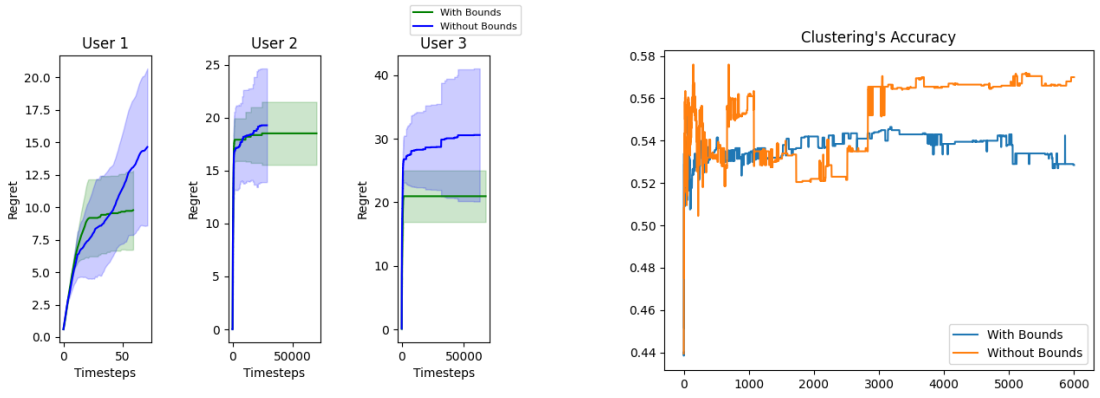
	$User\_type1$	$User\_type2$	$User\_type3$
$Pr(X = arm_1, Y = 1)$	1	0.8	0.9
$Pr(X = arm_2, Y = 1)$	0.8	1	0.5
$Pr(X = arm_3, Y = 1)$	0.2	0.4	1

3.2.1 Correct Constraints

Για να εξετάσουμε τις πλήρεις δυνατότητες του αλγορίθμου σε μια τέτοια περίπτωση, υποθέσαμε ότι ο διακομιστής είχε και πάλι κάποιες σωστές προηγούμενες γνώσεις για το γούστο του χρήστη. Συγκεκριμένα, κάθε φορά που ο διακομιστής επιλύει το πρόβλημα βελτιστοποίησης της ενότητας **3.2.3**, χρησιμοποιεί επίσης τους ακόλουθους περιορισμούς:

- $0.8 \leq \frac{Y^2(1)}{Y^3(1)} \leq 1.2$
- $0.45 \leq \frac{Y^3(2)}{Y^1(2)} \leq 0.8$
- $0.2 \leq \frac{Y^1(3)}{Y^2(3)} \leq 0.7$

Στη συνέχεια, η γραφική παράσταση του regret του κύριου αλγορίθμου της εργασίας που χρησιμοποιεί ορισμένες σωστές προηγούμενες πληροφορίες συγκρίνεται με αυτή του απλού αλγορίθμου UCB που δεν χρησιμοποιεί κανένα αιτιολογικό όριο κατά τη διάρκεια της εκμάθησης. Ενώ τα διαγράμματα regret της ενότητας **4.1** παρουσίαζαν μόνο τα σημεία δεδομένων των χρηστών που συγκεντρώθηκαν σωστά στο διακομιστή, θα πρέπει να ληφθεί υπόψη ότι για αυτήν την περίπτωση, σημεία δεδομένων εσφαλμένα ομαδοποιημένων χρηστών υπολογίστηκαν επίσης στο regret. Σε τελική ανάλυση, ακόμα κι αν ένας χρήστης ταξινομείται λάθος, εξακολουθεί να είναι δυνατό για το σύστημα και τους local clients να κάνουν τις σωστές προτάσεις.



(a) With and Without causal bounds

(b) Clustering Accuracy

Figure 6: Comparison of the algorithms with and without causal bounds, while using correct constraints for similar User-types.

Από το accuracy plot, μπορεί να παρατηρηθεί ότι σε αυτό το πείραμα ο αλγόριθμος αποτυγχάνει να ομαδοποιήσει σωστά τους χρήστες. Λαμβάνοντας υπόψη τις προτιμήσεις των χρηστών, μπορούμε να υποθέσουμε ότι οι χρήστες που συνήθως έχουν ομαδοποιηθεί με λάθος τρόπο ανήκουν είτε στον τύπο χρήστη 1 είτε σε 3, καθώς αυτές οι δύο ομάδες είναι παρόμοιες και μπορούν εύκολα να συγχέονται μεταξύ τους. Ωστόσο, ακόμη και σε αυτή την περίπτωση, οι τοπικοί πράκτορες στο πλευρό των πελατών, εξακολουθούν να καταφέρνουν να κάνουν τις σωστές συστάσεις που μεγιστοποιούν τη συνολική ανταμοιβή ή αντίστοιχα ελαχιστοποιούν ττο τελικό regret. Επιπλέον, μπορεί να παρατηρηθεί ότι για άλλη μια φορά η χρήση αιτιακών ορίων μπορεί να διευκολύνει τη σύγκλιση του αλγόριθμου πιο γρήγορα στην πραγματική και τελική του κατάσταση.

3.2.2 Σύγκριση με έναν αλγόριθμο UCB που δεν κάνει ομαδοποίηση

Προκειμένου να εξεταστεί η σημασία της ομαδοποίησης στον διακομιστή, κρίνεται απαραίτητη η σύγκριση του προτεινόμενου αλγόριθμου με τα αποτελέσματα που παρέχονται από ένα απλό UCB που αγνοεί την ύπαρξη διαφορετικών μοντέλων χρηστών στο σύστημα και δεν επιχειρεί καμία ομαδοποίηση. Σε αυτόν τον αλγόριθμο, ο διακομιστής παρέχει ένα σύνολο ανώτερων ορίων εμπιστοσύνης που είναι το ίδιο για όλους τους χρήστες, ανεξάρτητα από την ομάδα χρηστών που ανήκουν. Η σύγκριση των δύο αλγορίθμων φαίνεται στο σχήμα 21.

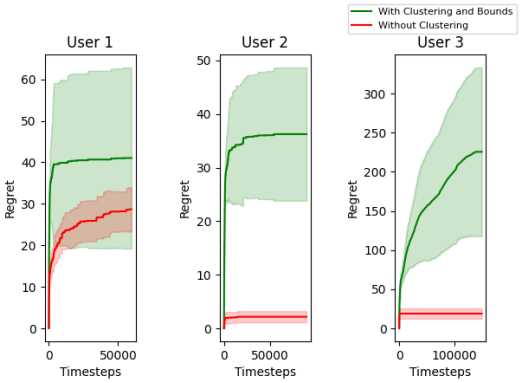


Figure 7: With and Without Clustering at the Server

Μπορεί να παρατηρηθεί ότι στην περίπτωση παρόμοιων τύπων χρηστών η ομαδοποίηση δεν είναι απαραίτητη δυνατότητα, χωρίς την οποία το σύστημα ενδέχεται να μην είναι σε θέση να κατανοήσει τις προτιμήσεις των χρηστών και να μην κάνει τις σωστές συστάσεις. Αντίθετα, η ομαδοποίηση μπορεί ακόμη και να εμποδίσει τη σύγκλιση του αλγορίθμου, ενώ η απόδοσή του μπορεί να μην είναι τόσο συνεπής όσο αυτή του απλού αλγορίθμου UCB χωρίς ομαδοποίηση (όπως φαίνεται από τη διακύμανση των γραφημάτων λύπης).

3.3 Γενική συζήτηση

Από τις παραπάνω προσομοιώσεις μπορεί να παρατηρηθεί ότι ο προτεινόμενος αλγόριθμος, 4, μπορεί να αποδώσει καλύτερα ή τουλάχιστον εξίσου καλό με το σημείο αναφοράς που εξετάστηκε (αλγόριθμος UCB που υλοποιεί ομαδοποίηση στο διακομιστή). Σε περιπτώσεις όπου χρησιμοποιούνται σωστοί περιορισμοί για τον υπολογισμό των αιτιακών ορίων, η απόδοση του αλγορίθμου βελτιώνεται δραστικά. Σε αντίθεση με το σημείο αναφοράς, το 4 είναι πιο συνεπές, όπως φαίνεται από τη στενή διακύμανση των διαγραμμάτων λύπης και διευκολύνει τη σύγκλιση του συστήματος πιο γρήγορα στην πραγματική του κατάσταση. Από την άλλη πλευρά, εάν η προηγούμενη γνώση που αποστέλλεται στον διακομιστή είναι λανθασμένη και διαταράσσει την ιδέα που έχει ο διακομιστής για τη σχέση μεταξύ των προτιμήσεων των χρηστών, η χρήση αιτιολογικών ορίων ενδέχεται να μην επιτρέψει στους τοπικούς πελάτες να μάθουν την αλήθεια των χρηστών. μοντέλο και έτσι να τους κάνει κατάλληλες συστάσεις. Ωστόσο, είναι πιθανό ακόμη και με τη χρήση εσφαλμένων περιορισμών να μην επηρεαστεί η απόδοση του αλγορίθμου, εφόσον ο διακομιστής εξακολουθεί να καταφέρνει να προσδιορίσει ποιο σκέλος προτιμά περισσότερο ο κάθε χρήστης.

Όσον αφορά την ακρίβεια ομαδοποίησης των αλγορίθμων, μπορεί να ειπωθεί ότι η εφαρμογή της μεθόδου ϵ-greedy είναι απαραίτητη προκειμένου ο διακομιστής να αποκτήσει μια πιο ολοκληρωμένη εικόνα των προτιμήσεων των χρηστών και έτσι να καταφέρει να τις ομαδοποιήσει με σωστό τρόπο. Στην πραγματικότητα, ακόμα κι αν η ακρίβεια ομαδοποίησης δεν φτάσει σε υψηλό ποσοστό, η ομαδοποίηση στον διακομιστή θα μπορούσε να θεωρηθεί σωστή. Για παράδειγμα, στην περίπτωση παρόμοιων τύπων χρηστών, παρόλο που η ακρίβεια ομαδοποίησης δεν μπορούσε να υπερβεί την τιμή των $0,6$, η ομαδοποίηση που έλαβε χώρα ήταν ακόμα σε θέση να βοηθήσει τους τοπικούς πελάτες να μάθουν τις προτιμήσεις των χρηστών και να τους προτείνει το σωστό σκέλος στο κάθε βήμα-βήμα.

Τέλος, μπορεί να παρατηρηθεί ότι υπάρχουν περιπτώσεις όπου η ίδια η ομαδοποίηση μπορεί να θεωρηθεί περιττή. Για παράδειγμα, εάν οι τύποι χρήστη ενός συστήματος έχουν αρκετά παρόμοιες προτιμήσεις προς τους βραχίονες, αρκεί απλώς να συνδυάσετε τα δεδομένα τους μαζί για να μάθουν οι τοπικοί πελάτες τις σωστές συστάσεις για κάθε χρήστη. Η προσπάθεια ομαδοποίησής τους, από την άλλη πλευρά, προσθέτει μια άλλη παράμετρο στο σύστημα που πρέπει να μάθει και ως αποτέλεσμα εμποδίζει τη μαθησιακή διαδικασία. Ωστόσο, εάν οι τύποι χρήστη είναι διαφορετικοί μεταξύ τους, η μη ομαδοποίηση τους στο πλάι του διακομιστή μπορεί να είναι επιζήμια για τη σύγκλιση του συστήματος στην πραγματική του κατάσταση. Δεδομένου ότι μια τέτοια πληροφορία είναι γενικά άγνωστη στον διακομιστή, η ομαδοποίηση των χρηστών σε ομάδες ανάλογα με το γούστο τους εγγυάται την απόδοση του αλγορίθμου.

4 Introduction

4.1 Previous Work - Current Innovation

Personalized recommendation systems have many applications ranging from therapy decision in medical applications [6], [7] and news article recommendation [11], [12] to online marketing [2], [15]. These systems generally involve multiple clients that collaborate with each other via a central server to learn optimal recommendation strategies that depend on their individual preferences. The objective is, e.g., to maximize the total click rates of the individual clients. This problem can be formulated as a Multi-Armed Bandit (MAB) problem where the arms represent different options available to the clients and the clients need to decide which arm to pull to maximize their accumulative reward. The MABs used by the clients to learn their local optimal decisions are coordinated by the central server so that system becomes a Federated Learning (FL) system. Compared to traditional FL that relies on existing data to learn a shared prediction model for all clients, here data are generated online and the optimal decisions are personalized to the individual client preferences. As in most FL literature, we assume that the clients do not share their private data with each other or the central server [14], [1].

Different variations of federated MAB problems have been proposed in recent literature. Initial attempts did not take into account the possible heterogeneity between the clients and proposed methods of learning one main global model for everyone. For example, [19], [17] develop methods to ensure efficient client-server communication and coordination such that the global bandit model is optimally found, even though both acknowledge the existence of different local models at each client. On the other hand, [11] and [3] assume that all clients act according to the same linear model and develop a UCB-type algorithm to allow the clients to collaboratively learn the best recommendations. More recently, personalized Federated MAB methods have also been developed to account for diversity among the clients. Specifically, [4] proposes a method to find an initial shared model that both current and new clients can use as an initialization to learn their local optimal decision. The work in [20] extends the method developed in [19] to balance both generalization (finding an optimal global bandit model) and personalization (defining a local bandit model per client). Lastly, [8] assumes different linear models for each client that are coupled together through global common parameters, and proposes a method based on federated learning.

A common limitation of MAB methods is that they generally require a large number of samples to find the optimal arm. This limitation becomes even more pronounced in personalized federated MAB problems where clients have different preferences and, therefore, form smaller populations that are more difficult to sample sufficiently. One way to address the sampling complexity of MAB methods is using Transfer Learning (TL)[[23]] and the knowledge gained from one "source" task to solve a new "target" learning problem; see, e.g., [16], [18]. In personalized federated MAB problems, TL can transfer information from one client type (source) to another (target), reducing in this way the sampling complexity of learning optimal recommendation strategies for all individual clients. However, in a system whose clients have varied preferences and transfer of information happens through the central server that does not have access to private client information, the source data available at the server and the target data at the clients will possibly be heterogeneous. Both the unknown information, [21], and this heterogeneity between the source and target data in TL problems, [10], are known to introduce bias in the learnt policies or value functions. To control the bias that can be caused due to non co-founder parameters, [13] proposes a UCB and a Thompson Sampling algorithm and addresses both linear and non-linear causal constrained MAB problems. Respectively, [22] proposed a causal bound constrained UCB algorithm to transfer information from an expert agent (source) to a learner agent (target) in an attempt to surpass the bias caused to users' different models.

In this paper, we propose a new UCB-type algorithm for personalized federated MAB

problems that employs causal bounds to transfer information across different client types with different private preferences. The goal is that clients in the same group collaborate to train a common model. More specifically, we define a client's type as a contextual variable that affects a client's reward function. We assume that this context is subject to a fixed distribution and is only known by the clients locally. Without this information, the central server cannot compute the true expected reward for each client type, e.g., the true click-through rate. Instead, it can compute causal bounds that provably contain the true reward. To transfer information across user types, we express the true reward of each arm as a linear model that combines the unknown rewards and context probabilities of the different user types that can be estimated using clustering. Since the true reward is bounded by the causal bounds, we finally obtain linear constraints on the unknown rewards for each arm and all different user types that provide new UCB-type bounds that are coupled across the different MABs, achieving in this way transfer across user types. A key feature of our algorithm is its ability to deal with errors in the clustering of users to different user groups. Such errors can result in user selection bias and, therefore, bias in the learning process. To address this challenge, we implement an ϵ-greedy method to enable fair exploration and improve clustering accuracy during learning.

To the best of our knowledge, causal Transfer Learning for personalized federated MAB problems has not been studied in the literature. Perhaps the most closely related work to the proposed method is [5]. In [5] users are assumed to belong to different groups of different models; for them to train their unique model, an iterative algorithm that alternates between estimating the cluster identities and minimizing the loss functions is suggested. Compared to [5], here users not only can collaboratively train their group's common model, but they can also transfer information to other user types accelerating their learning processes.

4.2 Structure of Diploma Thesis

The rest of the diploma thesis is organized as follows. In section 5 to 9, we analyze some theoretical and technical terms that are used in this project. Specifically, we present the definitions of recommendation systems, Federated Learning and Multi-armed Bandit problems that are necessary for the comprehension of the problem's structure. Moreover, we explain 2 of the project's key notions, causality and clustering. After the theoretical examination of the problem, we explain its formulation in section 10. Then, we analyse the proposed algorithm in section 11 both from the server's and the clients' perspective. Last but not least, we examine the algorithm's performance by conducting numerical experiments in section 12 that cover all the possible scenarios.

Part II
Theoretical Part

5 Recommendation Systems

A recommender or recommendation system constitutes a subclass of information filtering system, whose main goal is to predict the rating a user would give or the preference of a user towards an item. Recommendation systems are used in a variety of areas, with the most commonly recognised examples being playlist generators for movie, video and music services, product recommenders for online stores, or content recommenders for social media platforms. They are mainly used for providing both personalized content, in order to improve the on-site experience by creating dynamic recommendations for different kinds of audiences, and better product search experience, by categorising the product based on their features.

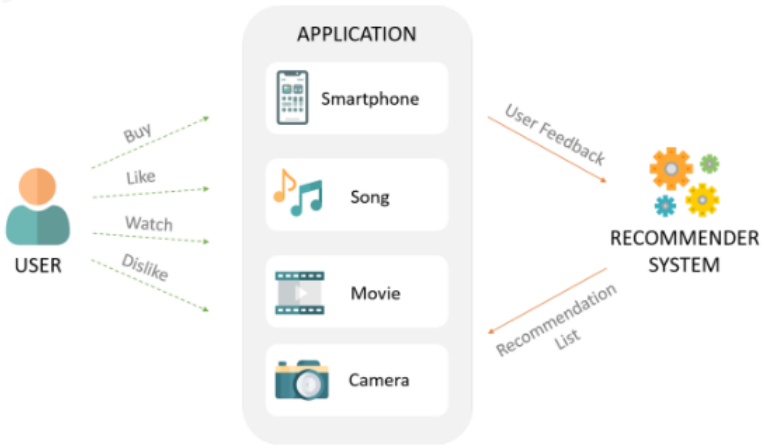


Figure 8: Illustration of the logical process of a recommendation system.

5.1 Approaches to the design of a recommendation system

One of the most common approaches for designing a recommendation system is **Content-Based Filtering**. Content-based filtering methods create a description of the system's items and a profile for the users. A user's profile relies not only on the user sign-in mechanism, but also on the user's interaction with the system. These kind of recommenders treat recommendation as a user-driven classification problem and learn a classifier for the user's preferences based on an item's features. These algorithms recommend items similar to those that a user liked in the past or is examining in the present. In particular, various candidate items are compared with items previously rated by the user, and the best-matching items are recommended.

The second most popular approach to the design of a recommendation system is **Collaborative Filtering**. The methods of this approach are based on the assumption that users who seemed to have similar preferences in the past will agree in the future, and that they will continue to like items similar to those they liked in the past. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself. However, this approach also has some major limitations. One of them is the problem of cold start, since there are not enough data for a user or an item new to the system, in order for an accurate recommendation to be made. Moreover, in many of the environments in which these systems are implemented, there are millions of users and products. As a result, a large amount of computation power is often necessary to calculate recommendations, and thus making scalability one of this approach main problems. Lastly, sparsity is a limitation that also need to be dealt with, since even active users will only have rated a small subset of the overall database.

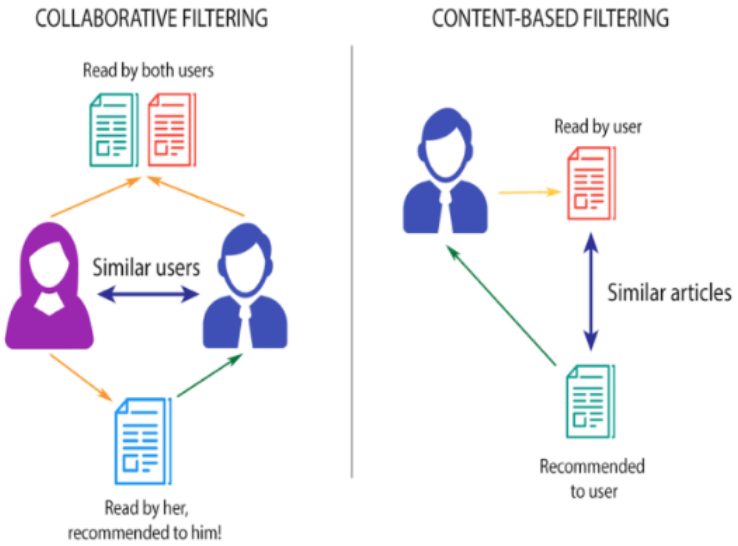


Figure 9: Principles of Collaborative and Content-based Filtering

Another approach to design the recommender system is to consider the recommendation problem as **reinforcement learning problem**. In this format, the user could be seen as the environment in which the agent, recommendation system, acts in order to receive a reward (e.g. a click or engagement by the user). While traditional learning techniques rely on supervised learning approaches that are less flexible, reinforcement learning recommendation techniques allow to potentially train models that can be optimized directly on metrics of engagement, and user interest. This aspect of reinforcement learning, meaning the fact that the models or policies can be learned by providing a reward to the recommendation agent, is of particular use in the area of recommender systems.

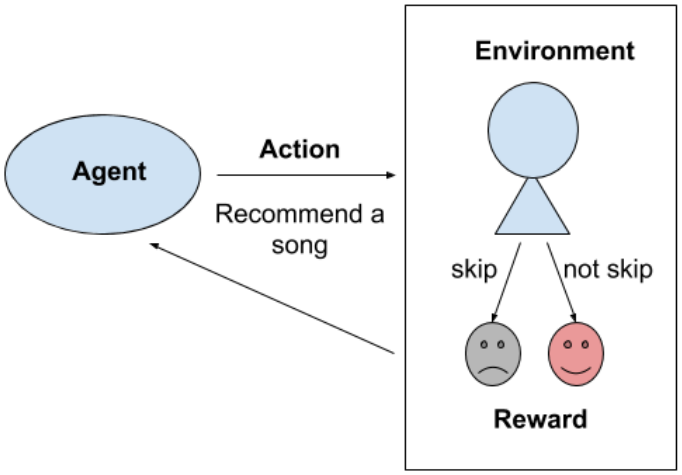


Figure 10: Example of Reinforcement Learning in Recommendation Systems

Last but not least, a recommendation system that is recently gaining more in popularity is the **Risk-aware recommender systems**. The majority of existing approaches to recommender systems focus on recommending the most relevant content to users using contextual information, and completely ignore the risk of disturbing the user with unwanted notifications. However, it is important to consider the risk of upsetting the user by pushing recommendations in certain circumstances, for instance, during a professional meeting, early morning, or late at night. Therefore, the performance of the recommender system depends in part on the degree to which it has incorporated the risk into the recommendation process.

6 Federated Learning

6.1 Introduction to Machine Learning

Artificial intelligence (AI), intelligence demonstrated by machines, is an academic term that was founded in 1956 and since then has experienced several waves of interest. One of the types of AI that has dominated the scientific research in the recent years is Machine Learning. Machine Learning is the field of understanding and building methods that learn from data and use the knowledge gained to improve on some set of tasks. The methods mentioned compute models based on sample data, called training data, in order to predict or make decisions and thus allowing software applications to become more accurate at estimating an outcome, without being explicitly programmed to do so.

Taking into account the way an algorithm learns in order to become more accurate to its predictions, machine learning can be categorized into three main approaches; Supervised Learning, Unsupervised Learning and Reinforcement Learning. Specifically, Supervised Learning refers to algorithms that build mathematical models based on a set of data that contains both the inputs and the desired outputs. On the other hand, Unsupervised Learning algorithms train on data that have no information on the desired output, and extract estimations based only on the inputs. Lastly, Reinforcement Learning is a method that teaches a machine to complete a multi-step process with clearly defined rules. It is used in environments where software agents ought to take actions so as to maximize some notion of cumulative reward.

Often, algorithms and theories used in Machine Learning are related to those met in other scientific fields like Data Mining, Statistics and Optimization. In more details, there are methods that can be applied to both ML and Data Mining problems, but while the first focuses on predictions and estimations based on already known aspects of the data, the latter targets the discovery of unknown properties in the data. Moreover, even though statistics and ML are quit related in terms of the methods used in both fields, they defer in their in their main objective; statistics' goal is to extract population inferences from a data sample, while ML is used to find general predictive patterns. Last but not least, ML is closely correlated to the field of Optimization, since many learning problems are formulated as minimization of some loss function on a training set of examples. As a matter of fact in the current thesis, that proposes an algorithm based on ML methods, the main goal is the minimization of a regret function.

In recent days, Machine Learning is used in a wide range of applications. The most well-known and studied example of the ML in use is the recommendation engine. In addition to that, other uses of ML include the following:

- **Customer relationship management software** that uses ML to identify the most important messages that need to be answered faster and to even suggest potential responses.
- **Business intelligence** that uses ML in their software to recognize important data points, patterns of data points and anomalies.
- **Self-driving cars** that via ML algorithms make it possible for a autonomous car to adapt at the current environment and alert the driver respectively.

Taking into account the wide range of application that Machine Learning can be used to, and its ability to easily identify trends and patterns in large volume of data without the demand for human intervention, it is no wonder that ML has gained so much in popularity in the recent years. However, ML comes with disadvantages as well. First and foremost, Machine learning projects are typically driven by data scientists, and thus the learning process requires expensive software infrastructure that can process a large amount of data in a short period of time. Furthermore, there is also the problem of machine learning bias.

Algorithms trained on data sets that exclude certain populations or contain errors can lead to inaccurate models that will fail or even become discriminatory.

6.2 Definition of Federated Learning

Federated or Collaborative Learning is a Machine Learning method that trains a decentralized model across multiple edge devices that varies from smartphones, to medical wearables, to IoT devices etc.They use their local data samples to collaboratively train a shared model, without however exchanging their private data or information with each other or with a central location. This approach differs from traditional centralized machine learning techniques where all the local data-points are uploaded to one server. It, therefor, allows personal information to remain in local sites, reducing possibility of personal data breaches.

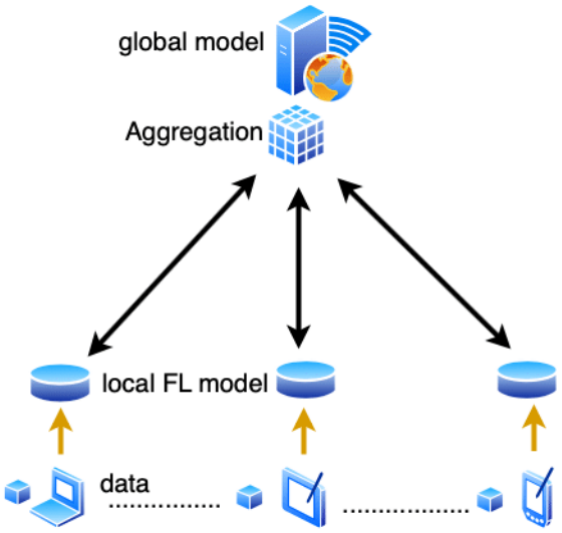


Figure 11: Structure of a Centralized Federated System

Since no data is shared at a global level, the general principle of Federated Learning consists in training local models on local data samples and then exchanging parameters (e.g. weights and biases of neural networks, upper confidence bounds, causal bounds etc.) between local devices and the central server to generate a global model common for all these devices. The algorithms used in Federated Learning rely on an iterative process that contains a set of clients-server interactions known as federated round. Each round begins with the current global model being transmitted to all participating clients. Afterwards, by producing new local data samples, clients develop a set of newly trained local models, which are then aggregated and processed by the server in order for an updated version of the global model to be created. Such a federated round can be summarized as follows:

- **Initialization of the system:** The server selects a machine learning model (e.g. confidence bounds) that is either pretrained on the central server or isn't trained at all. Then clients are activated and wait for the server to announce them the calculation tasks.
- **Distribution of global model:** The initial model created at the server's side is distributed to the clients
- **Local training:** Each client trains the model locally on their private data, according to a pre-specified fasion determined by the central server.

- **Reporting:** When locally trained, the updated models are sent back to the central server. The updates from all clients are averaged and aggregated into a single shared model, improving its accuracy.

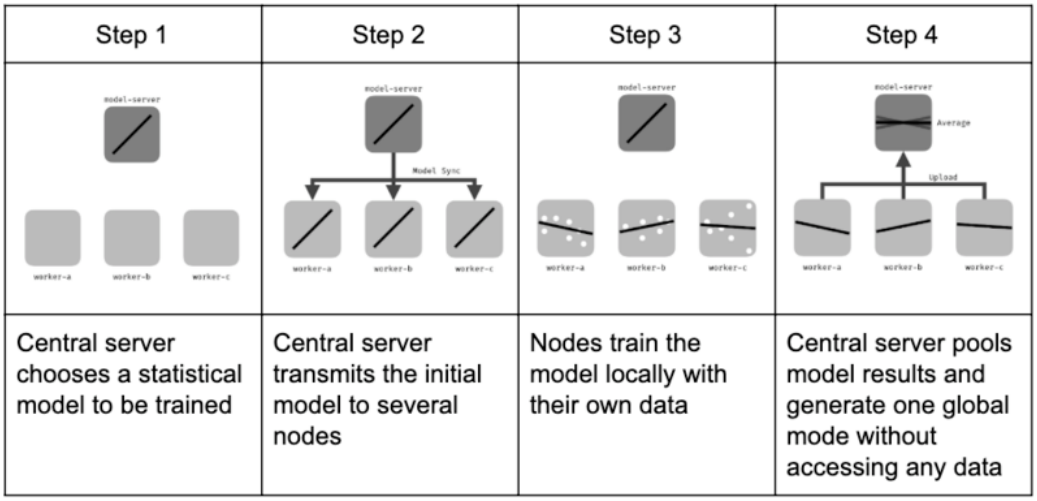


Figure 12: Federated Round

The federated Learning analysed thus far, is the centralized version, in which a central server is responsible for the orchestration of the algorithms' different steps and the coordination of all the participating clients. However, Federated Learning can also be decentralized. In Decentralized FL setting, the clients are able to coordinate themselves in order to obtain the global model. In this structure, due to the fact that model updates are exchanged only between interconnected clients without the orchestration of the central server, single point failures can be avoided.

6.3 Advantages and Disadvantages of Federated Learning

Federated learning is an emerging area in machine learning domain and it already provides significant benefits over traditional, centralized machine learning approaches. The main advantage of using FL methods is to ensure data privacy. Since no local data is exchanged or uploaded externally, it is more difficult to hack into it. On the contrary, in FL only machine learning parameters are exchanged, which can also be encrypted before sharing between learning rounds in order to extend secrecy. Moreover, Federated Learning is a method that is often used in cases where personalization is needed(application domains that involve a large set of heterogeneous clients). The central model delivers information on global patterns, and allows participating clients to adapt its outcomes on their peculiar preference. Federated Learning methodology can easily be adapted to generate two or more models at once in a multi-task learning framework. This version is called Heterogeneous federated learning.

Despite its innovative main idea, Federated Learning is far from perfect and actually faces a lot of technical limitations. To begin with, it is a method that is considered quite expensive, since frequent communication between clients and the server is required. This translates into high demand for storage capacity and large bandwidth and thus more investment requirements. In addition, even though data privacy is one of FL's main concerns, it cannot always be guaranteed. For instance, the fact that there are multiple devices for collecting and analyzing data, actually increases the attack surface and makes it harder for hacking attempts to be prevented. Also, despite the fact that only models, and not raw data, are communicated to the central server, models can possibly be reverse engineered to identify client data. Last but not least, lack of access to clients' information

and hiding training data can create problems during the learning process. For example, it could make it harder to identify unwanted biases entering the training process, or it might even allow attackers to alter the global model according to their will.

6.4 Federated Learning Applications

Undoubtedly, Federated Learning's key principal, which enables continual learning on end-user devices while ensuring data privacy and secrecy, has made it quite suitable for recent problems. Although there are many limitations that need to be surpassed and it is still being actively researched and improved, there already are applications in which FL is practically used.

1. **Healthcare:** Considering that protected health information can't be shared that easily due to recent regulations, Healthcare is one of those industries that can benefit from federated learning the most. In this way, masses of diverse data from different healthcare databases and devices can contribute to the development of AI models, while complying with regulations.

2. **Advertising:** Advertising as an industry, relies on each individual user's data in order to achieve personalization of the local models and make appropriate and suitable recommendations to the clients. However, nowadays users are becoming more concerned with their privacy and the amount of data that is shared and thus becoming public. Therefor, Federated Learning is a technique that can be applied in order for the clients' demands to be satisfied.

3. **Autonomous vehicles:** Since Federated Learning is capable of providing real-time predictions, the approach is used in developing autonomous cars. The information may include real-time updates on the roads and traffic conditions, enabling continuous learning and faster decision-making. This can provide a better and safer self-driving car experience. Although, at the moment, there is only research being conducted in the direction of FL being implemented in automotive industry, it is estimated that federated learning can reduce training time in wheel steering angle prediction in self-driving vehicles.

7 Multi-armed Bandit problem

The problem of a recommendation system is usually formulated as a multi-armed bandit problem (MAB). In those kind of problems a different set of resources (arms) must be allocated between various clients in a way that maximizes their expected gain, when each client's properties are only partially known at the time of allocation, and may become better understood as time passes or by allocating arms to the clients. During the learning process, each client provides a random reward from a probability distribution specific to that client. The objective of the system is to maximize the sum of rewards earned through a sequence of arm pulls. The crucial tradeoff a recommender faces at each trial is between "exploitation" of the arm that has the highest expected payoff and "exploration" to get more information about the expected payoffs of the other arms.

Considering one client, the MAB problem can be seen as a set of real distributions $R_1, \cdots, R_M$, each distribution being associated with the rewards delivered by one of the M arms.Let $\mu_1, \cdots, \mu_M$ be the mean values associated with these reward distributions. The client iteratively pulls one arm per round and observes the associated reward, based on the objective of maximizing the sum of the collected rewards. The bandit problem is formally equivalent to a one-state Markov decision process. The regret r after T rounds is defined as the expected difference between the reward sum associated with an optimal strategy and the sum of the collected rewards:

$$r = T\mu^* - \sum_{t=1}^{T} r_t$$

where $\mu^* = \max_k \mu_k$ and r_t the reward on round t.

8 Causality

Causality, also referred to as cause and effect, is influence by which one event, process, state, or object - the cause - contributes to the production of another event, process, state, or object - the effect - where the cause is partly responsible for the effect, and the effect is partly dependent on the cause. Generally, a process has many causes, which are also said to be causal factors for it, and all lie in its past, while an effect can in turn be a causal factor for many other effects, which all lie in its future. Causality is an abstraction that indicates how the world progresses, so basic a concept that it is more apt as an explanation of other concepts of progression than as something to be explained by others.

8.1 Types of Causes

Causes can sometimes be distinguished into three main types, as explained bellow:

- **Necessary causes:** A causal factor x can be characterised as a necessary cause of y if the presence of y necessarily implies the prior occurrence of x, while the presence of x does not guarantee the occurrence of y.
- **Sufficient causes:** A causal factor x can be characterised as a sufficient cause of y if the presence of x necessarily implies the subsequent occurrence of y, while the presence of y does not imply the prior occurrence of x.
- **Contributory causes:** A causal factor x can be characterised as a contributory cause of y if in a specific case was among several co-occurrent causes, but it cannot be considered neither necessary nor sufficient cause.

8.2 Causal Calculus

Generally causality can be formulated by various theories, like counterfactual or probabilistic ones. In this paper causality is examined in the form of causal calculus. When experimental interventions are infeasible or illegal, the derivation of a cause-and-effect relationship from observational studies must rest on some qualitative theoretical assumptions, usually expressed in the form of missing arrows in causal graphs such as Bayesian networks or path diagrams. The theory underlying these derivations relies on the distinction between conditional probabilities, as in $P(y|x)$, and interventional probabilities, as in $P(y|do(x))$.The theory of causal calculus - also known as do-calculus - permits one to infer interventional probabilities from conditional probabilities in causal Bayesian networks with unmeasured variables.

9 Clustering

Clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups. Each one of these groups is called cluster. It is a main task of exploratory data analysis, and a common technique for statistical data analysis, used in many fields, including machine learning that is this paper's main concern. Clustering itself is not one specific algorithm, but the general task to be solved, that can be achieved by various algorithms that differ significantly in their understanding of what constitutes a cluster and how to efficiently find them. Clustering can therefore be formulated as a multi-objective optimization problem. The appropriate clustering algorithm and parameter settings depend on the individual data set and intended use of the results. After all, clustering as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. Thus, It is often necessary to modify data preprocessing and model parameters until the result achieves the desired properties.

As it is already mentioned, there is not one singular definition of the term "cluster", which is practically one of the reasons why there are various clustering algorithms. While a cluster is mainly understood as a group of data object, different cluster objects can be employed by different researchers, with each one of them depending on a different algorithm. Typical cluster models include the following:

- **Connectivity models:** As the name suggests, these models are based on the notion that the data points closer in data space are more similar to each other than the data points lying farther away. These models can follow two approaches. In the first approach, they start with classifying all data points into separate clusters and then aggregating them as the distance decreases. In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. The distance function chosen in each application is a parameter that can be tuned according to the problem. While these models are very easy to interpret, they lack in scalability when handling big datasets. Examples of these models are hierarchical clustering algorithm and its variants.

- **Centroid models:** These are iterative clustering algorithms in which the notion of similarity is described by the closeness of a data point to the center of the clusters. K-Means clustering algorithm is a popular algorithm that falls into this category. In these models, the number of clusters should be given as an input to the algorithm applied, which makes it important to have prior knowledge of the dataset.

- **Distribution models:** These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution, e.g. Normal or Gaussian. A popular example of these models is Expectation-maximization algorithm which uses multivariate normal distributions. One of these models main disadvantage is that they often suffer from overfitting.

- **Density models:** These models search the data space for areas of varied density of data points in the data space. It isolates various different density regions and assign the data points within these regions in the same cluster.

9.1 k-Means Algorithm

In this paper, the algorithm that is used for clustering users according to their features is k-Means. k-Means clustering is a method of vector quantization, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as an identity of the cluster. This results in a partitioning of the data space into cells. Even though this problem is

computationally difficult (NP-hard), there are multiple efficient heuristic algorithms that converge quickly to a local optimal solution. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions. One of the biggest problem of the k-Means-type algorithms is that they require the number of clusters, k, to be specified in advance.

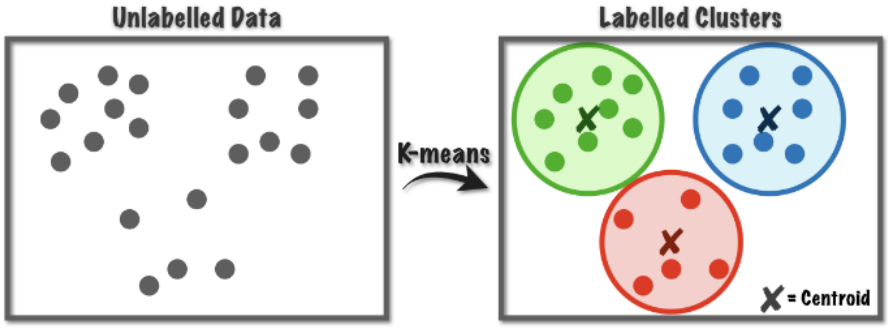


Figure 13: k-Means

The most common algorithm uses an iterative refinement technique, that is often referred to as "naïve k-means", because there exist much faster alternatives. Given an initial set of k centroids ($m_1^{(1)},\cdots,m_k^{(1)}$), which is often computed by doing random partition of the observations, the algorithm proceeds by alternating between two steps:

- **Assignment step:** Each observation is assigned to the cluster with the nearest centroid, e.g. with the least squared Euclidean distance.
- **Update step:** After the assignment of the observations, the clusters' centroids should be recalculated.

The algorithm converges, when the observations' assignments no longer change.

Part III
Experimental Part

(a) Client's side (b) Server's side

Figure 14: Causal graphs at the client's and server's side accordingly. U represents the user type, which affects both the decisions and rewards provided at the client's side, but is unobserved at the server's side.

10 Problem Formulation

We consider a recommendation system modeled with M arms, e.g., M possible recommendations, where the users belong to one of N different user types, according to a fixed distribution P_U. We assume a user-type u is defined as the type of user that has explored arm u and is aware of the fact that they like it. Both a user's type and their preferences towards the rest of the arms can be affected by their personal features and characteristics. For example in [9] a user's emotional state is monitored and used in order for the system to be able to suggest a suitable type of music. There can be various ways in which a user's feature affect their feedback to the system. For instance, there are cases in which it is assumed that a user's preference towards an arm is connected to their features through a linear model [11], [8]. In order to protect the users' privacy by keeping their characteristics hidden, and also not make assumptions as to what kind of model connects the users' features to their preferences (e.g. linear model), we consider the user-type as the only information that can be used during the learning process and we regard it as the cause of the users' preferences towards the arms.

As in every recommendation system, the goal is to maximize the overall reward, which in this case is equivalent to whether a user liked an arm recommended to them (with reward 1), or not (with reward 0). However, since we assume that the users have already explored an arm they like, their main goals are to determine their preferences towards the remaining arms and decide which of them could be their second optimal choice. We formulate the problem as a MAB defined by the tuple $\big(U, X, Y^U(X)\big)$, where $U \in \{1, \dots, N\}$ is a random variable that models the user-type, $X \in \{1, \dots, M\}$ is a random variable that indicates the selection of one of M arms, and $Y^u(x)$ is the reward function associated with arm $X = x$ given user-type $U = u$. Specifically, $Y^u(x)$ represents the probability of a user of type u liking arm x. At each time step, a user of type u is drawn independently from the distribution P_U. Based on the information provided by the server for that specific user, the local client pulls an arm from the pool and recommends it to the user. The incurred reward follows the user's preferences function Y^u.

Since the users can be clustered into groups according to their type, we design a federated learning framework that can allow similar users to collaboratively train their common preference model while leveraging the information gained by other user types. Under this formulation, it can be considered that for each user there is a local client-agent that makes recommendations to them, and that the whole learning process is supervised by a central server. From the server's perspective, the only available data are user-click data points, with the user's type remaining an unobserved confounder. As a result, on the server side, the only information that can be calculated with certainty are causal bounds on the arms' accumulated reward [22]. Therefore, the main goal is to design a method that helps the server calculate personalized causal bounds for each arm and for each user type and further to boost the learning process from the users' ends.

11 Algorithm Analysis

In this section we propose a UCB-based algorithm that takes into account the fact that the users belong to different user-types and uses this information to facilitate the learning process. We define the proposed method from both the client's and the server's perspective. Initially, we analyze the way the server uses the user-click data points sent by the clients, in order to cluster the users into different user groups. Afterwards, we describe the method for computing both confidence and causal bounds for each user-group, as a means of transferring learning both between users of the same type and different user-types respectively. Lastly, we explain the data generating process in the clients' side, which is mainly affected by the causal-UCB algorithm 4 that is implemented at every client.

11.1 Server's Side

During the learning process, the server only has access to data points of the form (u, a, r) gathered by the users, where a is the arm examined and r is the reward that is equal to either 1 or 0. Since no information on the user's type is provided, the only certain knowledge that can be computed in the server are causal bounds on the expected reward of an arm [22]. In this case, let $p_{xy} = \Pr(X = x, Y = y)$ be the probability of arm x having the reward y, which can be calculated using the data sent to server by the clients.

Consider $X \in \{1, \dots M\}$, $Y \in \{0, 1\}$. Then given p_{xy} either as an empirically estimated value or an a priori knowledge, the expected reward of an arm X without knowing users' specific types, $\mathbb{E}[Y|do(X)]$, can be bounded by:

$$\Pr(X = x, Y = 1) \leq \mathbb{E}[Y|do(X = x)] \leq 1 - \Pr(X = x, Y = 0) \tag{6}$$

11.1.1 Clustering at the server side

In order for the causal and upper confidence bounds that the server computes to apply to every user, clustering the users and separating them into groups of similar interests is essential. The algorithm that is used for this process, 3, is a heuristic that uses the data provided by the users, to group them into clusters according to their preferences. Specifically, the server computes a vector x for each user that represents their average reward for each arm. Then, with the use of the clustering algorithm KMeans, the server is able to cluster the users into N groups. This clustering is enough for the server to provide the users with upper confidence bounds and causal bounds, which are needed for the causal UCB-algorithm that will be implemented at the clients' side.

Moreover, in case the way the user types are defined, in this problem formulation, is an information known to the server, the algorithm 3 contains also a heuristic that determines which group, created by the KMeans, corresponds to which user-type. The key idea of this algorithm relies on the user-types' definition. Therefor, the labelling of the groups is affected by which arm has the largest reward at the center of each group, while making sure that no user-type is assigned to two different groups.

Algorithm 3 Clustering and Labeling Algorithm

Require: For each user u a vector x_u with their average reward for each arm

1: Use KMeans to separate users into N groups based on their vector x_u.
2: For each group i created, compute the Center of the group c_i, which portrays the average rewards per arm of the users that were assigned at the mentioned group.
3: Assign each group to a user type by solving the following linear programming problem:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{N} -c_i y_i \\
\text{subject to} \quad & \sum_{k=1}^{M} c_i[k] y_i[k] = 1, \qquad i = 1, \dots, N, \\
& \sum_{i=1}^{N} c_i[k] y_i[k] = 1, \qquad k = 1, \dots, M, \\
& y_i[k] \in [0,1], \quad i = 1, \dots, N \quad \text{and} \quad k = 1, \dots M
\end{aligned}$$

4: Let l_i be the user-type user i according to the server
5: **for** $i = 1, \dots, N$ **do**
6: $\quad l_i \leftarrow \arg\max_{k \in \{1, \dots M\}} y_i[k]$, where $i \in \{1, \dots N\}$
7: **end for**

11.1.2 Update Upper and Lower Confidence bounds for the arms' reward

Once the server receives a data point of the form (u, a, r) from a specific user, the first step is to update the upper and lower confidence bounds of the arm in the user-type that u is clustered to, according to the server.

Consider a data point (u, a, r) sent by the user u that according to the server belongs to user-type k. Then, if t_k is the number of total data collected for user-type k, $N_k[a]$ the number of data collected for user-type k concerning arm a and $r_k[a]$ the total reward collected thus far for arm a in user-type k, the ucb and lcb of $Y^k[a]$ can be updated as:

$$\text{UCB}(Y^k(a)) \leftarrow \frac{r_k[a]}{N_k[a]} + \sqrt{1.5 \frac{\log t_k}{N_k[a]}}, \tag{7}$$

$$\text{LCB}(Y^k(a)) \leftarrow \frac{r_k[a]}{N_k[a]} - \sqrt{1.5 \frac{\log t_k}{N_k[a]}}. \tag{8}$$

Even though both upper and lower confidence bounds are computed, the local clients need only the upper bound when they are to determine which arm they should recommend to the users at each time-step. However, the computation of the lower confidence bound is also necessary, since it constitutes an extra constraint that the server will use during the calculation of the causal bounds. Every time the upper confidence bound, of the reward of type k for a specific arm, change, its updated version will be sent to all the users that according to the server belong to user-type k.

11.1.3 Computation of Causal bounds for the arms' expected reward

If we take into consideration the existence of different user types in the system with distribution P_U, then the expected reward $\mathbb{E}$[Y—do(X)] can be expanded as:

$$\mathbb{E}[Y|do(X)] = \sum_{i=1}^{N} P_U^i \mathbb{E}[Y^k|do(X)]. \tag{9}$$

Once the confidence bounds of $Y^k(a)$ are found or updated, it is possible to acquire causal bounds for $\mathbb{E}[Y^i|do(X = a)]$, where $i \in \{1, \cdots, N\}\backslash\{k\}$ by using the causal bounds of $\mathbb{E}[Y|do(X = a)]$ in (6) and the analysis in (9).

Let UCausal(a) and LCausal(a) be the upper and lower causal bounds of $\mathbb{E}[Y|do(X = a)]$ as calculated in (6). Let P_U be the distribution of the user-types that the server has found through clustering, and UCB($Y^k(a)$), LCB($Y^k(a)$) the updated confidence bounds of arm a in user-type k. Then, upper causal bounds for the reward of a in user-types other than k can be computed by solving the following optimization problem.

$$\begin{aligned}
\text{minimize} \quad & -UCausal(Y^i(a)) \\
\text{subject to} \quad & \sum_{j=1}^{N} P(U = j) \cdot UCausal(Y^i(a)) \leq \text{UCausal}(a), \\
& \sum_{j=1}^{N} P(U = j) \cdot UCausal(Y^i(a)) \geq \text{LCausal}(a), \\
& UCausal(Y^i(a)) \in \left[\text{LCB}(Y^j(a)), \text{UCB}(Y^j(a))\right], \quad j = 1, \ldots, N
\end{aligned}$$

In the above optimization problem, $Y^i(a)$ represents the upper causal bound of the reward of arm a for user-type i. By minimizing the value of $-Y^i(a)$, we manage to obtain the largest possible value for this upper bound. In order to obtain an upper causal bound for all the user-types, it is necessary to solve this optimization problem N times for $i \in\in \{1, \cdots, N\}\backslash\{k\}$. If any prior knowledge concerning the user-types and their preferences is given (e.g $\frac{Y^1(a)}{Y^2(a)} \in [0.3, 0.6]$), it can be used as an extra constraint in the above optimization problem to help the causal bounds become tighter.

11.2 Client's Side

At the clients' level, it is assumed that each user is aware of their type, meaning that they know which arm they already like and what are the remaining ones that they need to further explore. As a result, during the learning process, each user will be presented with arms selected from a pool accustomed to their type, and at each time-step the arm selected will be the one with the best prospects of reward (the prospects of reward is based on the arms' upper bounds). The data points generated from this process will be used by the server in the clustering process of the users. In order to allow the server to correctly cluster the users and improve the clustering accuracy, it is essential that at the beginning of the algorithm a fair exploration of the arms is occurred. To address this challenge, we implement an ϵ-greedy method, in which ϵ is a tuning parameter that adjusts the algorithm's exploration for each user during the first steps of the simulation.

Algorithm 4 Causal bound constrained ϵ-greedy-UCB

Require: List of Users U, Pool of arms X, UCB bounds on the rewards per arm for each user type $Y^U(X)$, causal bounds on $Y^U(X)$ (both bounds are provided by the server), number of time steps T

$\epsilon = 0.1$
for $t = 1, \ldots, T$ **do**
  Choose randomly one user u from U.
  Let k be the user's true type.
  $a \leftarrow \arg\max_x \min\{\text{UCB}(x), \text{UCausal}(x)\}$, where $x \in X \backslash k$
  $flag = \text{random.choice}([1, 0], [\epsilon, 1 - \epsilon])$
  if $flag == 1$ **then**
    $a \leftarrow \text{random.choice}([X])$
  end if
  $r \leftarrow \text{random.choice}([1, 0], [1 - Y^k(a), Y^k(a)])$, where r the reward provided
  Send the data point (u, a, r) to the server.
end for

It was observed that without the incorporation of the ϵ-greedy method in the algorithm, the clustering accuracy at the server's side, even in cases where user-types varied from one another, could not improve enough during the learning process. Once the causal bounds converged, they dominated over the confidence bounds and forced the local clients always pick and recommend the same arm. When the causal bounds were correct that arm would be the optimal one, otherwise the sub-optimal. In both cases, however, users would not have a chance to explore the rest of the system's arms and thus the server would not have a more complete picture of all of their preferences. As a result, since the clustering is based on the server's estimation of the users' taste, its accuracy could not possibly be satisfying.

With the ϵ-greedy method, the local clients were allowed to explore all of the system's arms and create a better idea of the user's preferences. This way, the server was able to comprehend their taste and thus cluster them more accurately, at least in the cases that user-types were more district. Both the ϵ parameter and the time at which the ϵ-greedy method will stop being active, are tuning parameters that can be defined according to the examined simulation and problem.

12 Numerical Experiments and Results

Consider a news recommendation system whose users can be grouped into 3 user types ($N = 3$) and can choose articles from a pool of 3 arms ($M = 3$). Each arm represents a different category of articles and each user-type k is defined as the group of users that have explored arm k and know it is their optimal arm. The users' next goal is to explore the rest of their arms and learn which one is their second best choice.

It is assumed that there are 50 users signed in at the system and that they are separated into the 3 groups according to the distribution $P_{U_1} = 0.2$, $P_{U_2} = 0.3$ and $P_{U_3} = 0.5$. Before the learning process begins, the system collects a number of initial data-points per user, in order for the server to have an initialization of a clustering. Those data are also used for determining the initial values of the ucb bounds per user-type. Specifically, in the experiments conducted, each user provides five initial data that are collected based on the local policy that recommends the best arm for each user with probability 0.6 and one of the other two arms with probability 0.2 each.

After the initialization of both the clustering at the server and the upper and lower confidence bounds, the algorithm is executed for 200000 time-steps. At each time-step, T, one of the 50 users is randomly chosen and is presented with an arm according to the model that the server provided him with. Consider that the user that was peeked belonged to the user-type k and the arm that was given to him was arm a. Then, the regret of the user-type k, for which the best reward is μ^k, at time-step T is computed according to the following equation:

$$\text{regret}_T^k = T\mu^k - \sum_{t=1}^{T} Y^k(a) \tag{10}$$

In order for the results to be generally valid, the main simulation is conducted 25 times and the final regret that is presented represents the average regret of all the trials.

The results of the experiments are compared to those of two benchmarks. The first one still attempts to cluster the users in the server's side and train their model collaboratively, but ignores the existence of unknown information and unobserved confounders. As a result, no causal bounds are computed or used during the learning process, and thus no information gained from one group is transferred to another. The second benchmark does not take into account that the users might have different models and thus preferences, and uses federated learning to train one common model for all the users without attempting clustering or causal transferring.

In this section, there are two main experiments that are conducted. The first examines user-types that are quite different from one another and at each user-type the second optimal choice declines a lot from the last one. In the second experiment, the user-types have a lot of similarities between them, even though their true models are fundamentally different. Moreover, for each user-type the arms at the pools do not vary regarding their rewards.

12.1 User-types that differ from one another

In the beginning of the experiment, it is essential to define the true models that affect the users' rewards towards the system's arms. Initially, each one of the system's 50 users is assigned to one of the 3 user types, according to the distribution $P_{U_1} = 0.2$, $P_{U_2} = 0.3$ and $P_{U_3} = 0.5$. Then the user types' preferences towards the arms are represented by the probabilities shown in table 3. Therefor, the true model that is defined for each user type, is actually the probability of a user type k clicking on an arm a and thus providing it with a reward equal to 1.

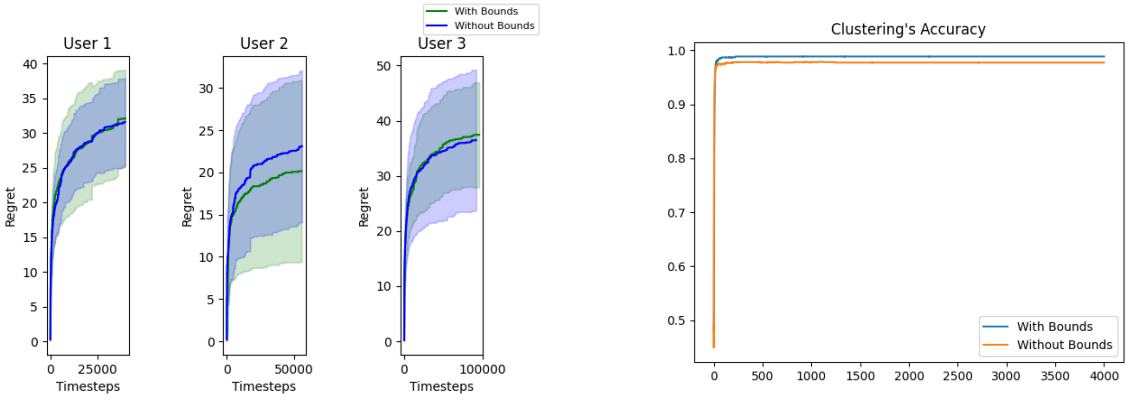
In the algorithm described above, it is possible that the server has previous knowledge on the models of each user type and, more specifically, on the relationship between the models. This knowledge could be considered as an a priori information provided by statistical or other surveys. Such information, could be used in the computation of the causal bounds as an extra constrain in the optimization problem described at the section **3.2.3**. In this section, we also examine whether this knowledge, if correct, is able to eventually accelerate the convergence of the causal bounds and thus of the models learnt at the clients, or if inaccurate would affect the algorithm's accuracy.

Table 3: Probability table that describes the preferences per user type.

	$User\_type1$	$User\_type2$	$User\_type3$
$Pr(X = arm_1, Y = 1)$	1	0.2	0.8
$Pr(X = arm_2, Y = 1)$	0.8	1	0.4
$Pr(X = arm_3, Y = 1)$	0.3	0.9	1

12.1.1 No constraints

Initially, in the formulation described above, the algorithm is implemented without using any extra information or constraints on the users' models. The paper's main algorithm is compared to that of a simple UCB that also implements clustering at the server, but does not compute or use any causal bounds while proposing arms to the clients. Their performance is assessed based on the regret plot, in which the regret is computed as shown in the equation 10, and the accuracy of the clustering at the server's side.



(a) With and Without causal bounds

(b) Clustering Accuracy

Figure 15: Comparison of the algorithms with and without causal bounds, while not using any constraints.

From this simulation, it can be observed that even though the use of causal bounds might accelerate the learning process for some users (in this case for users that belong in

type 2), it will generally not affect drastically the system's performance. It is known, that during the computation of the causal bounds, the confidence bounds are used as an extra constraint in the optimization problem. Therefor, the convergence of the causal bounds depend on the convergence of the confidence bounds, and as a result, when there is no other information that can be used in the optimization problem, the causal bounds cannot drastically improve the system's final results.

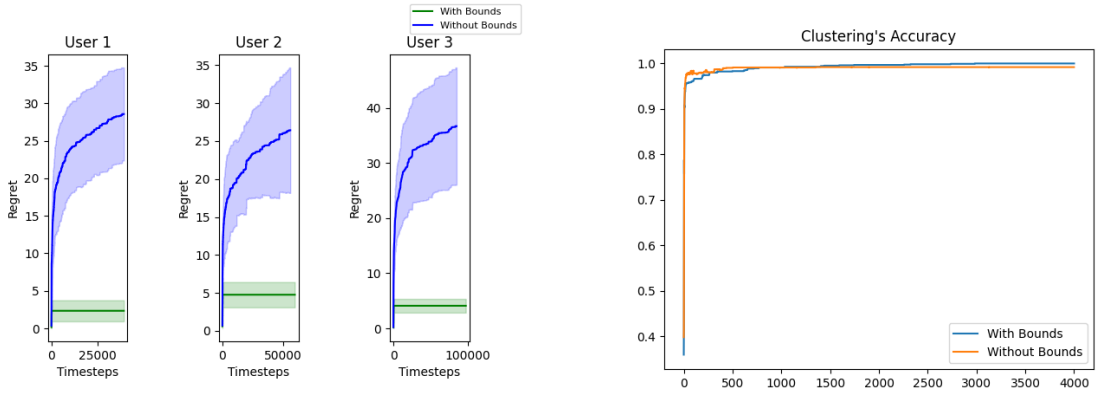
12.1.2 Correct constraints

While causal bounds facilitate some users learn their model faster, they generally do not make much of a difference in the algorithm's performance, since their improvement is interrelated to the improvement of the confidence bounds. Therefor it is evident that any correct prior information could help causal bounds become tighter and thus more helpful faster than the confidence bounds.

For the reason explained above, in this section we examine the case where the server has knowledge regarding the relationship between the users' models. Specifically, whenever the server solves the optimization problem described at the section **3.2.3**, it also uses the following constraints:

- $0.05 \leq \frac{Y^2(1)}{Y^3(1)} \leq 0.6$
- $0.3 \leq \frac{Y^3(2)}{Y^1(2)} \leq 0.7$
- $0.1 \leq \frac{Y^1(3)}{Y^2(3)} \leq 0.6$

Like before, the paper's algorithm is compared to the one that neither computes nor uses causal bounds.



(a) With and Without causal bounds

(b) Clustering Accuracy

Figure 16: Comparison of the algorithms with and without causal bounds, while using correct constraints.

In the case where prior information on the users' preferences is given to the server and is used as extra constraint in the optimization problem, it is evident that the use of causal bounds allows the system to converge almost immediately to its true state. It has been observed that when those extra constraints are active, causal bounds converge to their final value after only a few data-points. When the constraints provided are true, the bounds' final values are correct as well and approximate the users' preferences, thus allowing the local recommenders to make accurate suggestions to the users.

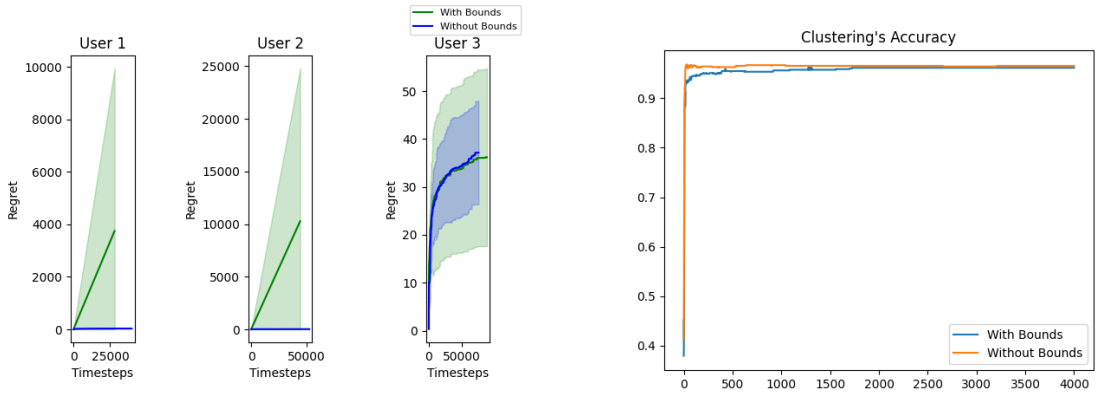
12.1.3 Wrong Constraints that affect the system's performance

Even though there are cases that correct constraints can potentially help the clients converge to their true model faster, the algorithm's robustness towards incorrect constraints is not guaranteed. The use of incorrect prior knowledge will possibly affect the values of the causal bounds and inevitably worsen the algorithms performance.

In this section, we examine the case in which prior knowledge transferred to the server is incorrect, and actually for the arms 2 and 3, represents the opposite relationship from the true one, as shown in the following constraints:

- $0.05 \leq \frac{Y^2(1)}{Y^3(1)} \leq 0.6$
- $0.3 \leq \frac{Y^1(2)}{Y^3(2)} \leq 0.7$
- $0.1 \leq \frac{Y^2(3)}{Y^1(3)} \leq 0.6$

The comparison between the algorithm with the wrong constraints and the one that does not use any causal bounds is shown in figure 17



(a) With and Without causal bounds (b) Clustering Accuracy

Figure 17: Comparison of the algorithms with and without causal bounds, while using incorrect constraints.

In this simulation, it is evident that incorrect constraints might disturb the algorithm's final results. When the true relationship between the users' preferences is not ensured, it is possible for the local recommenders to make suggestions to the users that are not suitable for their type, and thus allowing the algorithm's regret to keep increasing during all the learning process. The system's performance does not only change for the user types that are connected to the wrong information, in this case user types 1 and 2, but also for the rest of them, user type 3 for this simulation. Since the causal bound for the preference of a user type k for arm a is updated whenever another user-type provides information for that specific arm, if the interaction of at least one of the user types is altered, then the whole system is affected.

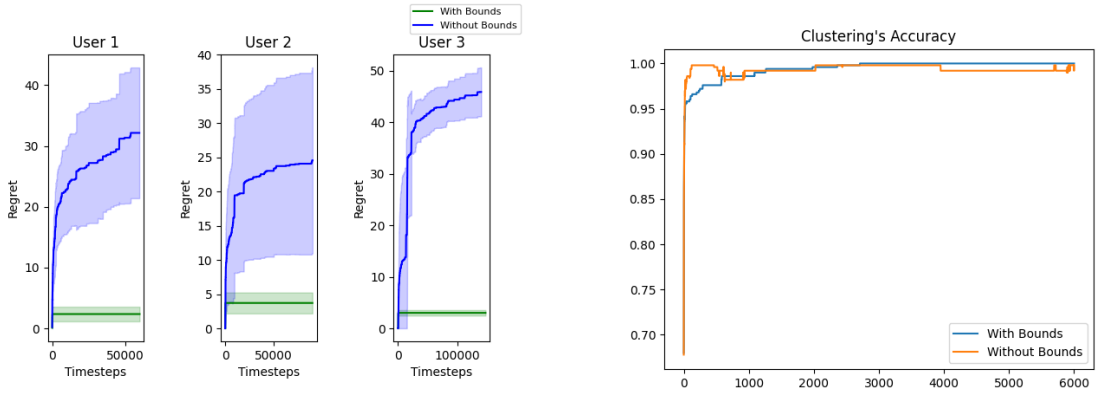
12.1.4 Wrong Constraints that do not affect the system's performance

The main goal of the algorithm is to determine which of the two remaining articles, that are not yet explored by the users, is liked the most by them. The minimum regret is then given if the local client recommends the most likable article to the users. Therefor, it is possible that even if the prior knowledge given to the server is incorrect, and as a result the bounds of the users' preferences towards the arm are wrong, the server still manages to learn the users' optimal arm. No matter what the server's estimation is (correct or not) for the actual rewards of the user-types for the arms, as long as it can still learn which arm is better than the other, the system will still succeed its main purpose.

In order to prove that such a case can exist, we simulated the formulation described above, and for the optimization problem solved by the server, we provided the following wrong constraints:

- $0.4 \leq \frac{Y^2(1)}{Y^3(1)} \leq 0.6$
- $0.6 \leq \frac{Y^3(2)}{Y^1(2)} \leq 0.7$
- $0.4 \leq \frac{Y^1(3)}{Y^2(3)} \leq 0.6$

As before, the paper's algorithm is compared to the one that does not take into account the existence of causality in the problem, and thus does not use any causal bounds.



(a) With and Without causal bounds (b) Clustering Accuracy

Figure 18: Comparison of the algorithms with and without causal bounds, while using incorrect constraints.

From figure 18, it can be observed that even though the server was provided with incorrect constraint regarding the users' preferences, the use of causal bounds is still quite helpful to the convergence of the system. Although the value of the reward that the server would expect ($E[Y^u|do(X)]$) is not correct, the server still manages to learn what the best suggestions for each user are.

12.1.5 Comparison with a UCB-algorithm that does no clustering

Thus far, in all the above experiments, it was assumed that the server new the existence of user-types, and even more, the actual number of the system's groups. Since the importance of clustering at the server might be questioned, we considered it appropriate to compare the proposed algorithm with the results provided by a simple UCB that ignores the existence of user-types in the system and does not attempt any clustering. In this algorithm, the server provides one set of upper confidence bounds that is the same for all the users, no matter which user-group they belong to. The comparison of the two algorithms is shown at figure 19.

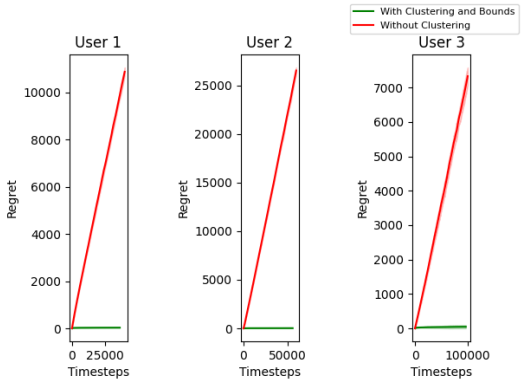


Figure 19: With and Without Clustering at the Server

It is perceptible that when no clustering takes place, since the users' preferences differ, combining their data to learn one common model cannot lead to the desired results. In this case for example, none of the local agents manages to learn the true model of the three user-types. Even though the server ignores the existence of user groups, the local agents keep making biased recommendations to the users according to their type. This biased and restricted collection of data-points, combined with the fact that the server does not acknowledge that the data-set is derived from different distributions, forces the system to converge to a wrong final state.

12.2 Similar User types

In the previous experiments, the user types that were defined were quite distinct and their preferences varied from one another. However, another interesting case that can be examined is when users of different types actually have similar taste. Like before, in the beginning of the experiment it is essential to define the true models that affect the users' rewards towards the system's arms. Initially, each one of the system's 50 users is assigned to one of the 3 user types, according to the same distribution (meaning $P_{U_1} = 0.2$, $P_{U_2} = 0.3$ and $P_{U_3} = 0.5$). Then the user types' preferences towards the arms are represented by the probabilities shown in table 4. As in all the previous experiments, whenever a user likes an article and clicks on it, they provide it with a reward equal to 1

Table 4: Probability table that describes the preferences per user type.

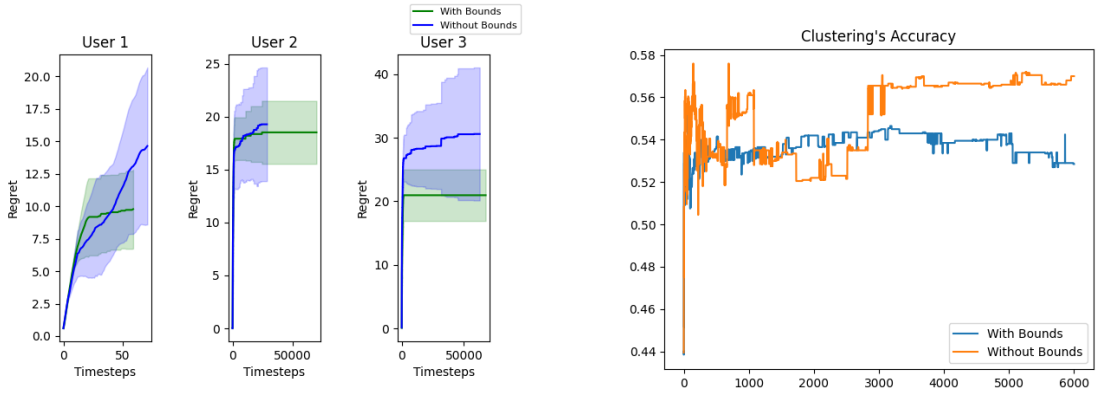
	$User\_type1$	$User\_type2$	$User\_type3$
$Pr(X = arm_1, Y = 1)$	1	0.8	0.9
$Pr(X = arm_2, Y = 1)$	0.8	1	0.5
$Pr(X = arm_3, Y = 1)$	0.2	0.4	1

12.2.1 Correct Constraints

In order to examine the full potentials of the algorithm in such a case, we assumed that the server again had some correct prior knowledge on the user's taste. Specifically, whenever the server solves the optimization problem of section **3.2.3**, it also utilizes the following constraints:

- $0.8 \leq \frac{Y^2(1)}{Y^3(1)} \leq 1.2$
- $0.45 \leq \frac{Y^3(2)}{Y^1(2)} \leq 0.8$
- $0.2 \leq \frac{Y^1(3)}{Y^2(3)} \leq 0.7$

Afterwards, the regret plot of the paper's main algorithm that uses some correct prior information is compared to that of the simple UCB algorithm that does not use any causal bounds during the learning. While the regret plots of section **4.1** presented only the data-points of the users that were clustered correctly at the server, it should be taken into account that for this case data-points of incorrectly clustered users were also accumulated in the regret. After all, even if a user might be clustered the wrong way, it is still possible for the system and the local clients to make the right suggestions.



(a) With and Without causal bounds

(b) Clustering Accuracy

Figure 20: Comparison of the algorithms with and without causal bounds, while using correct constraints for similar User-types.

From the accuracy plot, it can be observed that in this experiment the algorithm fails to cluster the users correctly. Taking into account the users' preferences, it can be assumed that the users that are mostly clustered the wrong way belong to either user-type 1 or 3, since those two groups are similar and can easily be confused for one another. However, even in this case, the local agents at the clients' side, still manage to make the right recommendations that maximize the overall reward, or respectively minimize the final regret. Moreover, it can be observed that once again the use of causal bounds can facilitate the algorithm converge faster to its true and final state.

12.2.2 Comparison with a UCB-algorithm that does no clustering

In order to examine the importance of clustering at the server, it is considered necessary to compare the proposed algorithm with the results provided by a simple UCB that ignores the existence of different user-models in the system and does not attempt any clustering. In this algorithm, the server provides one set of upper confidence bounds that is the same for all the users, no matter which user-group they belong to. The comparison of the two algorithms is shown at figure 21.

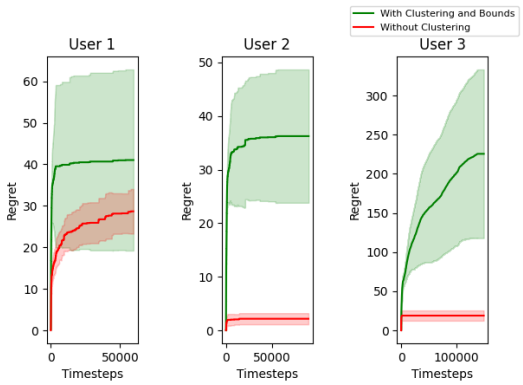


Figure 21: With and Without Clustering at the Server

It can be observed that in the case of similar user-types clustering is not a necessary feature, without which the system might not be able to understand the users' preferences and not make the correct recommendations. On the contrary, clustering can even hinder the algorithm's convergence, while its performance might not be as consistent as the one of the simple UCB-algorithm without clustering (as it can be seen from the variance of the regret plots).

12.3 General Discussion

From the above simulations it can be observed that the proposed algorithm, 4, can perform better or at least as good as the benchmark examined (UCB-algorithm that implements clustering at the server). In cases where correct constraints are used for the calculation of causal bounds, the algorithm's performance is drastically improved. In contrast to the benchmark, 4 is more consistent, as it can be seen from the tight variance of the regret plots, and facilitates the system converge faster to its true state. On the other hand, if the prior knowledge sent to the server is wrong, and disturbs the idea that the server has for the relationship between the users' preferences, the use of causal bounds might not allow the local clients to learn the users' true model and thus make suitable recommendations to them. However, it is possible that even with the use of incorrect constraints the algorithm's performance might not be affected, as long as the server still manages to determine which arm each user prefers the most.

Regarding the algorithms' clustering accuracy, it can be stated that the implementation of the ϵ-greedy method is essential in order for the server to obtain a more complete picture of the users' preferences and thus manage to group them together in a correct way. Actually, even if the clustering accuracy does not reach a high percentage, the clustering at the server could still be considered correct. For instance, in the case of similar user-types, even though the clustering accuracy could not exceed the value of 0.6, the clustering that occurred was still able to help the local clients learn the users' preferences and suggest them the right arm at each time-step.

Lastly, it can be observed that there are cases where clustering itself can be considered unnecessary. For instance, if the user-types of a system have quite similar preferences towards the arms, simply combining their data together is enough for the local clients to learn the correct recommendations for each user. Attempting to cluster them, on the other hand, adds another parameter to the system that needs to be learnt and as a result hinders the learning process. However, if the user-types are distinct from one another, not clustering them at the server's side can be detrimental for the system's convergence to its true state. Since such an information is generally unknown to the server, clustering the users to groups according to their taste guarantees the algorithm's performance.

13 Conclusion

In this thesis we proposed a Federated MAB algorithm for recommendation systems that take into account the users' individuality and unique preferences, while attempting to protect their privacy. The proposed algorithm, in most of the cases outperforms other simple UCB-type algorithms and manages to accelerate drastically the learning process. Theoretical analysis of the algorithm and more experimentation with the method and the definition of clustering require further study.

References

[1] Mohammed Aledhari, Rehma Razzak, Reza M Parizi, and Fahad Saeed. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 8:140699–140725, 2020.

[2] Wei Deng, Yong Shi, Zhengxin Chen, Wikil Kwak, and Huimin Tang. Recommender system for marketing optimization. *World Wide Web*, 23(3):1497–1517, 2020.

[3] Abhimanyu Dubey and AlexSandy' Pentland. Differentially-private federated linear bandits. *Advances in Neural Information Processing Systems*, 33:6003–6014, 2020.

[4] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[5] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020.

[6] Felix Gräßer, Stefanie Beckert, Denise Küster, Jochen Schmitt, Susanne Abraham, Hagen Malberg, and Sebastian Zaunseder. Therapy decision support based on recommender system methods. *Journal of healthcare engineering*, 2017, 2017.

[7] Ching-Hsien Hsu, Amir H Alavi, and Mianxiong Dong. Interactive personalized recommendation systems for human health, 2021.

[8] Ruiquan Huang, Weiqiang Wu, Jing Yang, and Cong Shen. Federated linear contextual bandits. *Advances in Neural Information Processing Systems*, 34, 2021.

[9] Tae-Yeun Kim, Hoon Ko, Sung-Hwan Kim, and Ho-Da Kim. Modeling of recommendation system based on emotional information and collaborative filtering. *Sensors*, 21(6):1997, 2021.

[10] Alessandro Lazaric and Marcello Restelli. Transfer from multiple mdps. *Advances in neural information processing systems*, 24, 2011.

[11] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[12] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40, 2010.

[13] Yangyi Lu, Amirhossein Meisami, Ambuj Tewari, and William Yan. Regret analysis of bandit problems with causal background knowledge. In *Conference on Uncertainty in Artificial Intelligence*, pages 141–150. PMLR, 2020.

[14] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[15] Loredana Mocean and Ciprian Marcel Pop. Marketing recommender systems: a new approach in digital economy. *Informatica Economica*, 16(4):142, 2012.

[16] Sudipan Saha and Tahir Ahmad. Federated transfer learning: concept and applications. *Intelligenza Artificiale*, 15(1):35–44, 2021.

[17] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 3:3, 2018.

[18] Tushar Semwal, Haofan Wang, and Chinnakotla Krishna Teja Reddy. Selective federated transfer learning using representation similarity. 2021.

[19] Chengshuai Shi and Cong Shen. Federated multi-armed bandits. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[20] Chengshuai Shi, Cong Shen, and Jing Yang. Federated multi-armed bandits with personalization. In *International Conference on Artificial Intelligence and Statistics*, pages 2917–2925. PMLR, 2021.

[21] Andrea C Skelly, Joseph R Dettori, and Erika D Brodt. Assessing bias: the importance of considering confounding. *Evidence-based spine-care journal*, 3(01):9–12, 2012.

[22] Junzhe Zhang and Elias Bareinboim. Transfer learning in multi-armed bandit: a causal approach. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1778–1780, 2017.

[23] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.