Constructive Recommendation

Paolo Dragone*
University of Trento, Italy
TIM-SKIL
paolo.dragone@unitn.it

ABSTRACT

Recommendation systems have seen a great burst in popularity and innovation in the past few years. However, both industrial applications and academic research have, been focussing on recommendation tasks involving objects taken from some fixed pool of available candidates (e.g. a database of products). A research direction that has remained unexplored involves the recommendation of objects that can be configured on the basis of the user preferences. We call this kind of recommendation tasks *constructive*, as the recommended objects are "built" from scratch to maximize the user satisfaction. Constructive tasks include recommendation of many types of objects that can be assembled from their components, such as PC configurations, recipes, travel plans or shared schedules to mention a few. Object synthesis usually involves solving a constrained optimization problem over a very large (or even infinite) domain of feasible configurations. For this reason, many of the existing recommendation techniques are not suitable in this setting, since they are based on the assumption of a finite and relatively small set of available objects. The objective of my research is to adapt or create new machine learning techniques suitable for constructive recommendation tasks.

ACM Reference format:
Paolo Dragone. 2017. Constructive Recommendation. In *Proceedings of ACM Conference on Recommender Systems - Doctoral Symposium, Bozen-Bolzano, Italy, August 2017 (RecSys'17),* 5 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Constructive recommendation tasks consist in recommending structured objects that can be created by assembling them from their components, such as PC configurations [11], travel plans [10] or layouts [3]. Each object is represented as a set of attributes, e.g. shape, size, color, i.e. Boolean or numeric variables that describe the physical properties of the object. The space of possible object configurations is exponential in the number of attributes, or infinite if some attributes are continuous variables. The possible configurations can also be constrained by arbitrary boolean formulas to limit the feasible space. We consider the problem of finding the best object for the user in this large combinatorial space of candidates. In this setting, classical recommendation techniques cannot be applied for several reasons. Most existing content-based and collaborative techniques are based on the assumption that available objects come from a fixed pool of existing ones that can be explicitly stored in a database. This is not the case in constructive recommendation, since the large domain of feasible objects is implicitly defined by the domains of the attributes and the imposed constraints. Another strong limitation of collaborative filtering in a constructive scenario is the structural scarcity of collaborative data, as having a very large domain of objects implies that most objects are only ever reviewed by a handful of users. For this reason we rather adopt an interactive paradigm in which every user provides (implicitly or explicitly) preference information to the system. The process of collecting preference statements from the user is called *preference elicitation*. Many preference elicitation techniques have been proposed over the years [4, 6, 13, 14] but most of them rely on the same assumption of a finite pool of candidates. An exception in this trend is the work from Teso et al. [11], which paved the way towards this research line. Conversely to its Bayesian competitors [4, 14], the system proposed by Teso et al. [11] is capable of scaling up to large constructive domains composed of several hundreds of possible configurations while still being competitive in terms of recommendation quality. Preference elicitation algorithms iteratively make recommendations to the user and ask for feedback. The objective is to maximize the users satisfaction while keeping them interested and engaged. This process can also be seen as an online preference learning problem. We proved that online techniques, such as Coactive Learning [9], can be used to learn user preferences and make recommendations in constructive settings [3, 10]. In the following sections I describe the two main components of a constructive recommendation system, namely the learning model and the inference solver. Next I outline some of my already accomplished works and lastly I highlight some research lines I will address in my PhD.

Algorithm 1 The Preference Perceptron algorithm [9].

```
procedure PreferencePerceptron(T)
    w^1 ← 0
    for t = 1, ..., T do
        Receive user query x^t
        y^t ← argmax_{y∈Y} ⟨w^t, φ(x^t, y)⟩
        Receive user feedback ȳ^t
        w^{t+1} ← w^t + φ(x^t, ȳ^t) − φ(x^t, y^t)
```

2 LEARNING MODELS

In constructive recommendation, user preferences are represented as a *utility* function $u : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that maps each object $y \in \mathcal{Y}$ to a real-valued score representing its level of user satisfaction. The user utility depends also on the "context" $x \in \mathcal{X}$, i.e. additional

*PD is a fellow of TIM-SKILL Trento and is supported by a TIM scholarship.

RecSys'17, Bozen-Bolzano, Italy

DOI: 10.1145/nnnnnnn.nnnnnnn

(implicit or explicit) side information available at recommendation time. The true user utility function $u^*(x, y)$ is unknown and never directly observed by the system. The system needs to collect some kind of preference information by interacting iteratively with the user. At each iteration t, as new preference information is acquired, the system can refine its estimate $u^t(x, y)$ of the true utility function. An assumption often made in the literature is to consider linear utility functions of the type $u^t(x, y) = \langle \boldsymbol{w}^t, \boldsymbol{\phi}(x, y) \rangle$ where $\boldsymbol{w}^t \in \mathbb{R}^m$ is the current estimate of the weight parameters of the utility model and $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^m$ is a feature map from the joint context-object space to an m-dimensional feature space. To estimate a utility function from the user preference feedback, we mostly focus on online preference learning algorithms, which are particularly suited to deal with constructive recommendation tasks. In our previous work, we often employed Coactive Learning algorithms [8, 9]. Coactive Learning is a framework for learning a utility function from the user manipulative feedback. After receiving a recommendation, the user is asked to provide a slightly improved version of the recommended object. Algorithm 2 shows the Preference Perceptron algorithm, which is one the simplest, but rather effective, Coactive Learning algorithms. At each iteration t, the algorithm receives a context $x^t \in \mathcal{X}$, which may include side information or some kind of user query. The system then makes a recommendation $y^t \in \mathcal{Y}$ by maximizing the current estimate of the user utility. The system then receives an improvement $\bar{y}^t \in \mathcal{Y}$ from the user, such that $u^*(x^t, \bar{y}^t) \geq u^*(x^t, y^t)$. Using the ranking information provided by the pair $(y^t, \bar{y}^t)$, the algorithm can update the current estimate of the weights.

The objective of the Preference Perceptron is to minimize the average *regret* of the system:

$$\mathrm{REG}^T = \frac{1}{T} \sum_{t=1}^{T} u^*(x^t, y_t^*) - u^*(x^t, y^t)$$

where $y_t^* = \mathrm{argmax}_{y \in \mathcal{Y}} u^*(x^t, y)$ is the optimal object for the user in context x^t. In Coactive Learning [9] the user is assumed to behave according to the following α-informative feedback model:

$$u^*(x^t, \bar{y}^t) - u^*(x^t, y^t) = \alpha(u^*(x^t, y_t^*) - u^*(x^t, y^t)) - \xi^t$$

This model requires the user feedback $\bar{y}^t$ to improve y^t of a fraction α of the regret, modulo a slack ξ^t. When the user behaves according to this model, the average regret of the Preference Perceptron is guaranteed to decrease as $O(1/\sqrt{T})$. In particular, the average regret is upper-bounded by (proof in [9]):

$$\mathrm{REG}^T \leq \frac{2R\|\boldsymbol{w}^*\|}{\alpha\sqrt{T}} + \frac{1}{T} \sum_{t=1}^{T} \xi^t \tag{1}$$

where $\boldsymbol{w}^*$ are the true weights of the user and R is the radius of the smallest ball enclosing the feature space, i.e. $\|\boldsymbol{\phi}(x, y)\| \leq R$.

3 OBJECT SYNTHESIS

The second component needed to build a constructive recommendation system is an inference procedure that, given the current utility model, generates new objects with high utility. In the Preference Perceptron, the recommendation step consists in finding the object with highest utility in the space $\mathcal{Y}$ of feasible configurations:

$$y^t = \mathrm{argmax}_{y \in \mathcal{Y}} \langle \boldsymbol{w}^t, \boldsymbol{\phi}(x^t, y) \rangle$$

As overviewed in Section 1, the configuration space $\mathcal{Y}$ is a combinatorial space defined by the domains of the attributes and the imposed constraints. Attributes may be boolean or numerical (integer or real) variables. Making inference over this space equates to solving a constrained optimization problem. The difficulty of the inference problem greatly depends on the type of constraints and features used. In the simplest case, when constraints and features are linear functions of the attributes, inference is a mixed integer linear program (MILP). While being NP-hard in the general case, modern solvers can find exact solutions to large problems in reasonable time. In many cases, however, exact solutions are too computationally expensive to find and thus approximation techniques may be used. The feasibility of non-linear optimization inference problems for constructive recommendation is still an open research question.

Algorithm 2 The Critiquing Preference Perceptron algorithm.

procedure CritiquingPerceptron
  $\boldsymbol{w}^1 \leftarrow 0$
  for $t = 1, \ldots, T$ **do**
    Receive query x^t from the user
    $y^t \leftarrow \mathrm{argmax}_{y \in \mathcal{Y}} \langle \boldsymbol{w}^t, \boldsymbol{\phi}(x^t, y) \rangle$
    User provides improvement $\bar{y}^t$
    if NeedCritique$(x^t, y^t, \bar{y}^t)$ **then**
      Receive critique ρ from the user
      $\boldsymbol{\phi}^t \leftarrow \boldsymbol{\phi}^t \circ [\rho]$
      $\boldsymbol{w}^t \leftarrow \boldsymbol{w}^t \circ [0]$
    $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \boldsymbol{\phi}^t(x^t, \bar{y}^t) - \boldsymbol{\phi}^t(x^t, y^t)$
    $\boldsymbol{\phi}^{t+1} \leftarrow \boldsymbol{\phi}^t$

4 INITIAL RESULTS

In this section I point out some of the results I have already found in my work. I first describe an extension of Coactive Learning to allow feature elicitation in constructive domains, called Coactive Critiquing. Secondly, I detail an applicative constructive recommendation system, using Coactive Learning, to recommend layouts, i.e. 2D arrangements of objects.

4.1 Coactive Critiquing

Feature selection is an important but quite daunting task. Particularly so in constructive recommendation systems. In general, relevant preference criteria (i.e. features) may differ from user to user. Engineering all the relevant features for all users is expensive and error prone. Moreover, learning and solving inference in a high-dimensional feature space requires more data and more computing time. In this work, we proposed a procedure to elicit relevant features as additional user feedback. We modified the Preference Perceptron algorithm to allow the system to ask *critiques* to the user. Critiques are "explanations" that are asked when the system receives ambiguous improvements. Given an example pair $(y^t, \bar{y}^t)$, a critique is a function discriminating $\bar{y}^t$ from y^t.

Algorithm 2 lists our Critiquing Perceptron algorithm. At each iteration t, after receiving the improvement $\bar{y}^t$ the system has the option of asking the user to provide a critique for the example. To decide whether a critique is needed, we check if the example is consistent with all the previous ones, i.e. whether there exists a weight vector $\boldsymbol{w}$ that describes all the ranking examples. If this

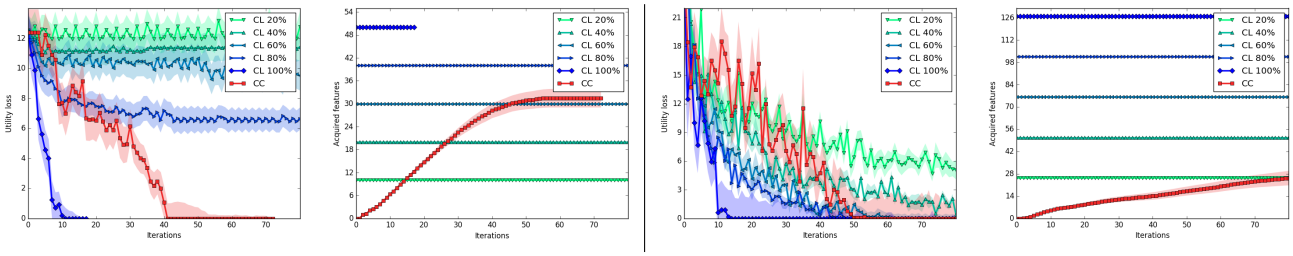

Figure 1: Regret and acquired features for the synthetic problem (left) and the trip planning problem (right) in Coactive Critiquing [10]. Best viewed in color.

check fails, it means that the current feature space cannot represent all the collected examples. When this happens we ask the user for a critique of the last example, which we then use to enlarge the feature vector with a new feature. In this way, the feature space grows only when necessary, keeping the learned model simple and inference time low.

We found that the Critiquing Perceptron enjoys a regret upper-bound analogous to that of the Preference Perceptron (Eq 1). For a α-informative user the average regret of the Critiquing Perceptron is upper-bounded by (proof in [10]):

$$\text{REG}^T \leq \frac{2R\|\boldsymbol{w}^*\|}{\alpha\sqrt{T}} + \frac{1}{T}\sum_{t=1}^{T}(\xi^t + \eta^t)$$

where

$$\eta^t = \langle \boldsymbol{w}^*, \boldsymbol{\phi}^*(x^t, \bar{y}^t) - \boldsymbol{\phi}^*(x^t, y^t)\rangle - \langle \boldsymbol{w}^*, \boldsymbol{\phi}^t(x^t, \bar{y}^t) - \boldsymbol{\phi}^t(x^t, y^t)\rangle$$

is the amount of utility gain the Critiquing Perceptron "misses out" by having only a subset $\boldsymbol{\phi}^t$ of the relevant user features $\boldsymbol{\phi}^*$.

We also tested our algorithm on two constructive settings, a synthetic problem and a realistic problem involving trip plan recommendations. We compared the Critiquing Perceptron (CC) with the standard Preference Perceptron (CL) using different percentages of user features. The former starts with only basic features and elicits more during the interaction, whereas the latter starts with a percentage of additional features on top of the basic ones but does not elicit more. As we can see in Figure 1, in both settings the Critiquing Perceptron is able to rapidly decrease the regret as CL 60% or 80% while eliciting only a fraction of the total user features.

4.2 Constructive Layout Synthesis

In this work, we presented a constructive recommendation system to recommend layouts, i.e. 2D arrangements of objects. Examples of recommendation problems involving layout synthesis are recommendation of furniture arrangements, space division, or urban planning. This system is able to learn the "style" of a designer or an architect and recommend new configurations on the basis of their preferences. Coactive Learning is particularly suited in this setting because we imagine to be able to learn from designers sketches as improvements to recommended configurations. We instantiated this system on a furniture arrangement task, in which the goal of the system is to learn to arrange tables in a room. The contexts x include the shape of the room and the number of tables, whereas the attributes of y consist in the coordinates and the sizes of all the tables in the room. Features $\boldsymbol{\phi}(x, y)$ include different high-level properties of the table arrangement, such as the maximum and minimum distances between the tables and the maximum and minimum distances between the tables and the walls. Several constraints are imposed to ensure the tables fit into the room and they do not overlap.

The difficulty of the inference problem grows with the number of tables, as more variables and constraints are added for each table. When the difficulty of the problem is too high to be solved in a small amount of time, we employ an approximation heuristic consisting in setting a time cutoff on the solver and retaining the best solution found in that time. We tested this system with 6, 8, and 10 tables with exact inference, and 10 tables with approximate inference. We can see from the plots on the left in Figure 2 that the regret follows the same descending pattern for all settings with exact inference, with minor differences as the problem complexity grows. The running times, however, become quickly unfeasible for the case with 10 tables. The approximate version, instead, has predictable running time for the price of a small loss in recommendation quality.

We also evaluate this setting qualitatively by showing how the system can learn to furnish different type of rooms depending on the user preferences. We selected two users with different goals, one who wants to furnish a café and another who wants to furnish an office. We run our algorithm on these users and we show, in the right block of Figure 2, different stages of the two elicitation processes. The initial configuration is random in both cases. In the intermediate configuration we can already see some emerging patterns, for instance the choice of mostly 1×1 tables in the café. In the final configurations, the café has all 1×1 positioned along the walls, as one might expect in a café, while the office contains mostly 1×2 desks positioned in a regular fashion across the room.

5 RESEARCH DIRECTIONS

Constructive recommendation brings up a wide range of interesting research questions, starting from the selection of the best learning model, to the difficulty of constrained optimization on large domains. My research is focused on addressing these issues with the aim of developing scalable constructive recommendation algorithms and applications. In the following, I outline some of the problems that I am planning to address during my PhD.

5.1 Online preference learning

Learning algorithms used in constructive recommendation systems should be fast and scalable to large volumes of data, as typical of

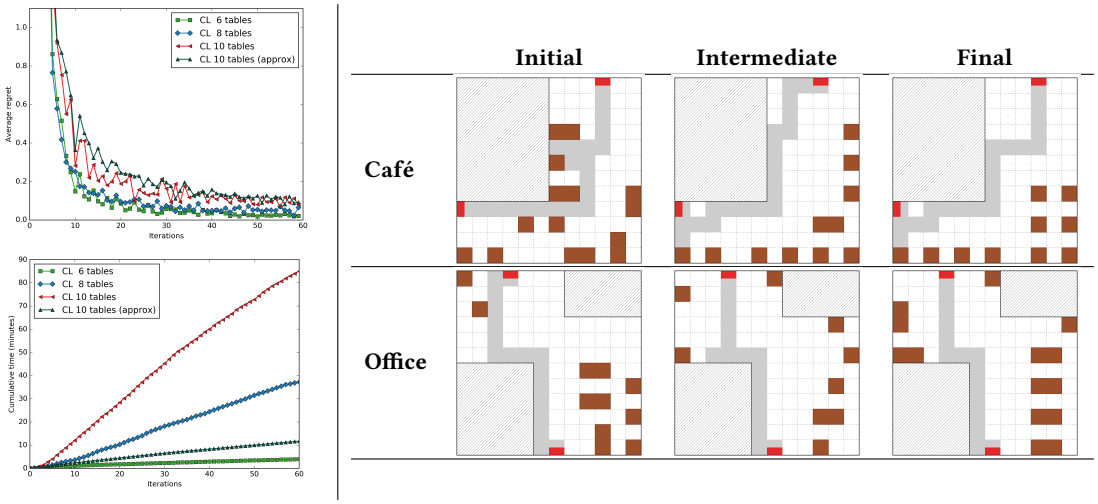

Figure 2: Left: average regret (top) cumulative running time (bottom) of the layout synthesis system [3]; Right: Two examples of configurations built by our system when used by users who are furnishing a café or an office. The figure shows different stages of algorithm while learning the user preferences. The tables are colored in brown and the doors in red. An automatically generated path between the doors is showed in grey the walls are represented with black striped sections. Best viewed in colors.

modern recommendation systems. Online preference learning algorithms are well suited candidates. There is, however, little work in online preference learning when it comes to defining the right interaction protocol with the user and the best policy to select the recommendation maximizing both the satisfaction of the user and the information gained by the feedback. Both these aspects are crucial in the constructive scenario. Furthermore, in the case of Coactive Learning, while this framework is well suited in some scenarios, manipulative feedback may not be easy to acquire from the user in many constructive tasks. An alternative to the coactive feedback is the choice set feedback, a well studied type of user interaction in the literature of algorithmic decision theory and preference elicitation [5, 7, 12]. Choice set feedback is often deemed less effort demanding for the user [4]. In a recent work, we developed an online algorithm, specifically designed for constructive tasks, to learn from choice set feedback. Differently from Coactive Learning, the use of choice set feedback brings up the problem of how to select recommendation sets. In this work, we developed a theoretically-driven selection strategy and we produced empirical results favouring our approach over existing preference elicitation techniques both in recommendation quality and running times. Future work in this direction include: [i] increase expressiveness of our preference models by employing long-standing preference elicitation tools, such as GAI utility functions and CP-nets [1]; [ii] generalize local feedback on portions of the objects (a.k.a. "sketches") by exploiting structural properties, e.g. generalize improvements over a single room to the entire house in layout synthesis tasks [3].

5.2 Feature selection and elicitation

As pointed out in Section 4.1, feature selection is a crucial aspect in constructive recommendation systems and Coactive Critiquing is one possible approach to learn relevant user features by eliciting them as additional critiquing feedback. While being quite effective, this approach requires further user involvement in providing critiques, which may be too cognitively expensive. An alternative that we are developing is to automatically learn new features directly from the user improvements or some other source of high-level feedback. Other possible approaches that we are investigating include reducing the space of relevant features via a sparsifying L_1 norm or by exploiting structural properties of the features, such as their *inheritance* [2].

5.3 Inference over combinatorial domains

Recommendations in constructive domains are selected by solving some kind of constrained optimization problem. As detailed in Section 3, the inference problem is a MILP problem when constraints and features are linear in the attributes. With increasing domain complexity solving inference exactly becomes impractical. As in Section 4.2 [3], a viable option is relying on some kind of approximate inference. However, we are still looking for some more reliable approximation technique with some guarantee on the final recommendation quality, instead of the somewhat unreliable solving time cutoff. Another interesting research question in this direction is whether we can go beyond linear programs and address inference in quadratically constrained domains. This would allow us to describe domains involving areas and euclidean distances, such as layout synthesis [3].

6 CONCLUSION

In this abstract I presented the topic of Constructive Recommendation, a framework to recommend objects created from scratch on the basis of the user preferences. I described the two main components of a constructive recommendation system, namely the learning model and the object synthesis. I outlined some of my accomplished work, and I highlighted some unexplored research direction that I will address in my PhD.

REFERENCES

[1] Craig Boutilier, Fahiem Bacchus, and Ronen I Brafman. 2001. UCP-networks: A directed graphical representation of conditional utilities. In *UAI*. 56–64.

[2] Hugh Chipman. 1996. Bayesian variable selection with related predictors. *CJS* 24, 1 (1996), 17–36.

[3] Paolo Dragone, Luca Erculiani, Maria Teresa Chietera, Stefano Teso, and Andrea Passerini. 2016. Constructive Layout Synthesis via Coactive Learning. In *Constructive Machine Learning workshop, NIPS*.

[4] Shengbo Guo and Scott Sanner. 2010. Real-time multiattribute Bayesian preference elicitation with pairwise comparison queries. In *AISTATS*. 289–296.

[5] Jordan J Louviere, David A Hensher, and Joffre D Swait. 2000. *Stated choice methods: analysis and applications*. Cambridge University Press.

[6] Gabriella Pigozzi, Alexis Tsoukiàs, and Paolo Viappiani. 2016. Preferences in artificial intelligence. *Ann. Math. Artif. Intell.* 77, 3-4 (2016), 361–401.

[7] Pearl Pu and Li Chen. 2009. User-involved preference elicitation for product search and recommender systems. *AI magazine* 29, 4 (2009), 93.

[8] Karthik Raman, Thorsten Joachims, Pannaga Shivaswamy, and Tobias Schnabel. 2013. Stable Coactive Learning via Perturbation.. In *ICML (3)*. 837–845.

[9] Pannaga Shivaswamy and Thorsten Joachims. 2015. Coactive Learning. *JAIR* 53 (2015), 1–40.

[10] Stefano Teso, Paolo Dragone, and Andrea Passerini. 2017. Coactive Critiquing: Elicitation of Preferences and Features. In *AAAI*.

[11] Stefano Teso, Andrea Passerini, and Paolo Viappiani. 2016. Constructive Preference Elicitation by Setwise Max-Margin Learning. In *IJCAI*. 2067–2073.

[12] Olivier Toubia, John R Hauser, and Duncan I Simester. 2004. Polyhedral methods for adaptive choice-based conjoint analysis. *Journal of Marketing Research* 41, 1 (2004), 116–131.

[13] Paolo Viappiani and Craig Boutilier. 2009. Regret-based optimal recommendation sets in conversational recommender systems. In *RecSys*. ACM, 101–108.

[14] Paolo Viappiani and Craig Boutilier. 2010. Optimal Bayesian recommendation sets and myopically optimal choice query sets. In *NIPS*. 2352–2360.