Semi-supervised Learning for Discrete Choice Models

Jie Yang, Sergey Shebalov, and Diego Klabjan

***Abstract*—We introduce a semi-supervised discrete choice model with algorithmic approaches to estimate choice models when relatively few requests have actual preferences but the majority only have the choice sets. Two classic semi-supervised learning algorithms, the expectation maximization algorithm and the cluster-and-label algorithm, have been adapted to our choice modeling problem setting. We also develop two new algorithms based on the cluster-and-label algorithm. The new algorithms use the Bayesian Information Criterion to evaluate a clustering setting to automatically generate new clusters out of existing clusters and adjust the number of clusters. Two computational studies focusing on travel demand forecasting (i.e. a hotel booking case and a large-scale airline itinerary shopping case) are presented to evaluate the prediction accuracy and computational effort of the proposed algorithms. Algorithmic recommendations are rendered under various scenarios based on the hotel booking case while economic insights are derived based on the itinerary shopping case.**

***Index Terms*—Semi-supervised learning; Discrete choice models; Travel demand forecasting**

I. Introduction

AIRLINES and hotels are trying to encourage travelers to book service directly through their own channels. For example, Lufthansa is looking to make a stand by charging an extra $18 for every ticket issued via a global distribution system (GDS) — the technology behind the booking systems used by travel agents and online travel agencies (Economist, 2015). GDS providers whose biggest asset is the data deluge are facing more intense competition. Confronting the industry challenges, GDS solutions have to understand better travelers' behavior and preference, to predict travel demand and market attractiveness, and subsequently monetize such findings, and thus prevent a corner overtaking.

The travel industry creates lots of data. Yet, traditional travel demand forecasting studies rely either on collecting travelers' responses from designed survey experiments, or on reconstituting choice sets by adding inferred choice alternatives to stated preferences. For example, Carrier (2008) combined observed flight booking data with fare rules and seat availability data to reconstitute the choice set of each booking. Mottini and Acuna-Agost (2017) used booking data matched with large data source coming from search logs (i.e. origin, destination and dates). Both approaches, however have their deficiencies. Conducting survey experiments is time consuming and labor intensive. It is hard to guarantee that a respondent's decision making process in a survey (stated preference) is consistent with that in a real booking environment. In addition, the designed choice sets, in some cases (e.g. flight booking, hotel booking, etc.) are more restrictive comparing to a real environment. Reconstituting choice sets is better but limited to simplified rules and assumptions. In summary, both approaches cannot reflect reality.

Nowadays GDS providers can store large data pertaining to received requests (either for air travel or hotel), and returned services (e.g. itineraries a traveler sees when booking). The actual bookings are captured by separate reservation systems that are not integrated with request and service information systems. In order to develop a discrete choice model (DCM), the actual preferences are needed.

We employ a semi-supervised approach to derive the preferences. We first assume that if there is an itinerary in the choice set dominating all other itineraries with respect to the fare and deviation from the request parameters such as the departure time and the elapsed time, then such an itinerary would be the preferred choice. This strategy creates requests with preferred choices. For all remaining requests we employ semi-supervised techniques to infer the preferred choices. We refer to the unmatched requests as unlabeled data and the matched requests with a responding itinerary as labeled data. Previous studies concentrate on choice modeling with only labeled data but simply utilizing limited labeled data may lead to bias. Leaving out unlabeled data is wasteful as the unlabeled data also captures travelers' preferences (e.g. to book an itinerary, travelers are often required to state their preferred airline, cabin, departure and arrival times, etc.). In some situations, unlabeled data could offer information to segment travelers and potentially prevent bias. In order to leverage the value of unlabeled data, we consider semi-supervised learning (SSL) which lies between supervised and unsupervised learning. In supervised learning, there is a known, fixed set of categories and category-labeled training data (i.e. a set of data to discover potential utility functions) used to induce a classification function. In contrast, unsupervised learning applies to unlabeled training data. Both supervised and unsupervised learning have been widely applied to transportation problems. For example, Zhang and Xie (2008) applied support vector machine which is a supervised learning method to travel mode choice modeling. Vlahogianni et al. (2008) developed a multilayer strategy that integrated the k-means algorithm to cluster traffic patterns. To the best of our knowledge, an SSL framework has not yet been

Jie Yang is with the Department of Civil and Environmental Engineering, Northwestern University, Evanston, IL, 60201 USA e-mail: jieyang2011@u.northwestern.edu.

Sergey Shebalov is with the Sabre Airline Solutions, Southlake, TX, 76092 USA e-mail: sergey.shebalov@sabre.com.

Diego Klabjan is with the Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 60201 USA e-mail: d-klabjan@northwestern.edu.

applied in choice modeling.

In machine learning, the SSL algorithms have been widely used to improve prediction accuracy in classification problems with also unlabeled data. The SSL algorithms usually make use of the smoothness, cluster, and manifold assumptions, and can be roughly categorized into five categories: (1) self-training; (2) SSL with generative models; (3) semi-supervised support vector machines; (4) transductive SSL with graphs; (5) SSL with committees (Zhu, 2008; Peng et al., 2015). We refer to Zhu (2008) and Chapelle et al. (2006) for details about SSL algorithms. The SSL algorithms have also been used in transportation research. Liu et al. (2016) evaluated the performance of two graph-based SSL algorithms for driver distraction detection which is considered as a binary classification problem. By using unlabeled data, their SSL methods improved the detection accuracy.

Generative models make the assumption that both labeled and unlabeled requests come from the same parametric model (Schwenker and Trentin, 2014). The SSL-DCM problem can be considered as a generative model problem and could be solved by traditional algorithm frameworks such as the expectation maximization (EM) algorithm which was first presented by Dempster et al. (1977) who brought together and formalized many of the commonalities of previously suggested iterative techniques for likelihood maximization with missing data. The cluster-and-label (CL) algorithm which is a heuristic model finds the label using labeled data within each cluster, and assigns labels to unlabeled requests within each cluster. Demiriz et al. (1999) related an unsupervised clustering method, labeled each cluster with class membership, and simultaneously optimized the misclassification error of the resulting clusters.

Besides the classic SSL algorithms such as the EM framework and CL applied to SSL-DCM, we introduce two new methods X-cluster-and-label-1 (XCL1) and X-cluster-and-label-2 (XCL2) based on the X-means algorithm (Pelleg and Moore, 2000). These algorithms have never been applied in the context of SSL and they have been designed to cope with the problem of setting the hyper parameter for the number of clusters required by CL. X-means is a method which includes model estimation inside the cluster partitioning procedure. It can efficiently search the space of cluster locations and number of clusters to optimize the Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC). Pelleg and Moore (2000) discovered that this technique could reveal the true number of classes in the underlying distribution and provide a fast, statistically founded estimation. In the XCL1 and XCL2 algorithms, we search partitions based on the entire data set and select the best result which optimizes the BIC of the labeled data set. An adapted BIC formula is derived to consider the maximized log-likelihood, the number of features and the number of requests.

As mentioned earlier, it is often difficult for transportation planners to collect a solid data set. Hence, discrete choice models with incomplete data have recently raised interest in transportation problems. Vulcano et al. (2010) developed a maximum likelihood estimation algorithm to use a variation of the EM method accounting for unobservable data with no-purchase outcomes in airline revenue management. Newman et al. (2012) applied the EM method to estimate the market share of one alternative that was not observed to be chosen in the estimation data set. Their case study aimed at demonstrating the consistency and potential viability of the methodology; significance levels of coefficient estimates are not reported and a more thorough evaluation of this needs to be conducted (Newman et al., 2012). Another type of incomplete data is called "choice-restricted" where there are unobserved choices in the choice set. Lindsey et al. (2013) only observed acceptance decisions but no alternatives. They used a supplementary sample (an unbiased sample with only features) to augment the likelihood function. All these works are different from our work. Vulcano et al. (2010) and Newman et al. (2012) solved the problem which has to only infer no-purchase alternatives. Lindsey et al. (2013) solved the problem when the data set only comes with the chosen choice. But in our problem, we do not infer the no-purchase alternatives and we have both chosen and unchosen choices in the labeled data. In addition, we have a rich set of unlabeled data. For details of discrete choice models, we refer to McFadden (1978), Ben-Akiva and Bierlaire (1999) and Koppelman and Sethi (2000).

Heterogeneity effects have also been widely studied in choice model. Generally, there are three approaches: (1) imposing a probabilistic prior on coefficients; (2) segmenting based on exogenous latent variables; (3) undertaking an endogenous segmentation (Bhat, 1997; Yasmin et al., 2014; Molesworth and Koo, 2016). Bhat (1997) pointed out that the second approach to capturing systematic heterogeneity assumes the existence of a fixed, finite number of mutually-exclusive market segments (each individual can belong to one and only one segment). The third approach, however jointly determines the number of segments, the assignment of individuals to segments, and segment-specific choice model parameters. In the third approach, each observation or individual is assigned to different clusters or segments based on probabilities. One of our algorithms assumes a fixed number of segments which is aligned with the second type of algorithms. The remaining two algorithms of ours use choice model based BIC criterion inside the clustering process to find the optimal number of clusters automatically which resembles the third type of algorithms. Depaire et al. (2008) applied a model-based clustering technique (i.e. latent class clustering) to segment traffic accident data. Mohamed et al. (2013) implemented cluster-based regression choice models (i.e. latent class and K-means) to investigate pedestrian injury severity outcomes in two different cases. In their case study, the latent class model did not work well in one case study as it allocated 90% of the dataset to the first two clusters regardless of the selected number of clusters (Mohamed et al., 2013). In addition to our contributions in the semi-supervised setting, the automatic partitioning algorithms make more general contributions to the transportation field.

In the computational experiments, we first apply a semi-supervised ranked-ordered logit (ROL) model on a hotel booking case to explore the capability and accuracy of the SSL-DCM algorithms developed herein (adapted EM, CL, XCL1 and XCL2). The hotel case is completely labeled and we use

it to evaluate the predictive power of our models. The ROL model was introduced in the literature by Beggs et al. (1981) and has also been used in transportation studies (Podgorski and Kockelman, 2006; Calfee et al., 2001). The ROL model can be transformed into a series of multinomial logit (MNL) models: an MNL model for the most preferred item; another MNL model for the second-ranked item to be preferred over all items except the one ranked first, and so on (Fok et al., 2012). In choice models, based on normal distributions, the top-N metric has been applied (Mottini and Acuna-Agost, 2017; Lhéritier et al., 2018). Since the ROL model in the hotel case requires a measure capturing rank predictions, we use four measures: log-likelihood, Kendall's τ, position difference and reciprocal rank difference to achieve the evaluation target. In the second case which herein motivated the entire work, we apply the SSL-DCM algorithms to an airline itinerary shopping data set. The heuristic we use to find labeled booking data is novel and only specific to this case study. Our contributions are summarized as follows:

- We mathematically define the semi-supervised discrete choice problem. To the best of our knowledge, no prior studies have been focused on this topic.
- We adapt two classic SSL algorithms to the semi-supervised discrete choice problem.
- We design two new algorithms (not previously known in the SSL context) which include model estimation inside the partitioning procedures.
- Stochastic gradient descent method is used in both computational cases which provides an alternative to solving large-scale choice model problems.
- For the two case studies, we design different metrics, show that the proposed algorithms have a good prediction accuracy and also provide recommendations of applying the algorithms. Economic insights are also derived from the real air travel case. The choice environment of hotel and air travel is also well suited in other transportation situations such as rail, ride sharing, bicycle sharing, etc.

II. Methodologies

A. Problem Setting

In the SSL-DCM problem, the input consists of a labeled data set D^L and an unlabeled data set D^U:

$$D^L = \{(X_1, L_1), (X_2, L_2), ..., (X_n, L_n)\}$$

$$D^U = \{(X_{n+1}, L_{n+1}), (X_{n+2}, L_{n+2}), ..., (X_{n+m}, L_{n+m})\},$$

where we assume that vectors $L_{n+1} = ... = L_{n+m} = \mathbf{0}$ and $eL_1 = ... = eL_m = 1$ (Here $e = (1, ..., 1)$.) The latter binary encodes the only selection in the choice set.

We have:

$$D^{U^*} = \{(X_{n+1}, L^*_{n+1}), (X_{n+2}, L^*_{n+2}), ..., (X_{n+m}, L^*_{n+m})\}$$

indicate that D^U has been assigned "soft-labels." We denote $I = \{1, 2, ..., (n+m)\}$. We also have labeled requests X^L and unlabeled requests X^U:

$$X^L = \{X_1, X_2, ..., X_n\}$$

$$X^U = \{X_{n+1}, X_{n+2}, ..., X_{n+m}\}.$$

Given set x, denote by $|x|$ the cardinality of x. In the traditional SSL problem, we have X_i defined as a feature vector and $L_i \in \{1, 2, ..., M\}$ where L_i is a class variable. In the SSL-DCM problem, we define the choice set $X_i = \{s_{i1}, s_{i2}, ..., s_{i|X_i|}\}$ where $s_{ij} \in \mathcal{R}^d$ is a feature vector and $|X_i|$ is the number of choices in X_i, and $L_i = \{\delta_{i1}, \delta_{i2}, ..., \delta_{i|X_i|}\}$ as a set of binary indicators encoding selection. We also assume a feature split $s_{ij} = (s^{ISF}_{ij}, s^{MSF}_{ij})$ exists where s^{ISF}_{ij} is a vector of individual or request specific features (e.g. respondent's income, gender and age group, etc.), and s^{MSF}_{ij} is a vector of alternative specific features and/or interactive features (e.g. one itinerary's departure time, respondent's income interacting with itinerary's cabin class). So we have $s^{ISF}_{ij} = v_i$ for every $j = 1, 2, ..., |X_i|$. Later we use $V = \{v_1, v_2, ..., v_{n+m}\}$. In $\{L_1, L_2, ..., L_n\}$ and $\{L^*_{n+1}, L^*_{n+2}, ..., L^*_{n+m}\}$, we have $\sum_{j=1}^{|X_i|} \delta_{ij} = 1$ while in $\{L_{n+1}, L_{n+2}, ..., L_{n+m}\}$, $\sum_{j=1}^{|X_i|} \delta_{ij} = 0$ holds.

We note that each feature in v^s should be correlated with some features from s^{MSF}_{ij} as otherwise such features can be dropped from v without effecting selections.

The EM algorithm iteratively solves discrete choice models with temporary labeled requests and assigns "soft-labels" to unlabeled requests until convergence. In each iteration, temporary labeled requests are changed. The CL algorithm finds a partition of X^L with a given number of clusters, then assigns labels to unlabeled requests using discrete choice models trained within each cluster, and eventually solves one model with all labeled requests. The XCL1 and XCL2 algorithms which have not yet been proposed for SSL start clustering with no fixed number of clusters, and develop partitions automatically. After the partitions have been found, the remaining steps are the same as in the CL algorithm.

B. EM Algorithm

Let Θ be a coefficient vector. Then the utility reads $U_{ij} = s_{ij}\Theta$ and the probability of choosing choice j for request i takes the classic multinomial logit form (McFadden, 1978):

$$Prob(\delta_{ij} = 1) = \frac{e^{s_{ij}\Theta}}{\sum_{k=1}^{|X_i|} e^{s_{ik}\Theta}}.$$

Let $H = \{H_1, H_2, ..., H_Q\}$ be the set of all possible $\{L^*_{n+1}, L^*_{n+2}, ..., L^*_{n+m}\}$ for unlabeled requests, with $Q = \prod_{i=n+1}^{n+m} |X_i|$. Also let $q()$ be the distribution function for hidden labels and denote t as the iteration number and let $f(q, \Theta) = \sum_{l=1}^{Q} q(H_l) \log(\frac{P(D, H_l|\Theta)}{q(H_l)})$.

In the E-step we compute $q(H_l)$ based on current Θ^t and find the "soft labels" for unlabeled data. This requires simple calculation based on utilities $U_i j$. Then we find Θ^{t+1} which maximizes $f(q, \Theta)$. This is the M-step and it gradually increases the lower bound of $\log(P(D|\Theta))$ until convergence. This step is the classic discrete choice utility coefficients computation algorithm based on maximum likelihood.

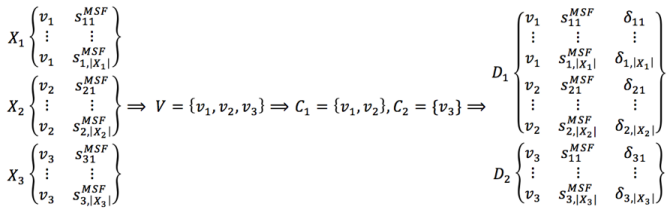


Fig. 1: An illustrative example for choice model clustering

C. CL Algorithm

In the CL, XCL1 and XCL algorithms, denote $K = \{1, 2, ..., |K|\}$ as the index set of the clusters. Let $C = \{C_1, C_2, ..., C_{|K|}\}$ be a set of clusters where C_k is a subset of I for any k in K. Let c^k be the centroid vector for cluster C_k. Given an arbitrary subset $C_k \in V$, we denote by D_k the matching subset corresponding to C_k in $D^L \cup D^U$. Let $l(C_k)$ be the number of labeled requests in cluster C_k.

When applying the CL algorithm in choice models, we consider the individual specific features as the clustering features. An example is presented in Figure 1 to illustrate the clustering process. Suppose we have three requests, X_1, X_2 and X_3. From them, we draw the individual specific feature vectors as $V = \{v_1, v_2, v_3\}$ and partition on V. In this example, requests X_1 and X_2 have been clustered together and X_3 is a standalone cluster. Then based on L_1, L_2 and L_3, we generate the clustered data set D_1 and D_2 for choice modeling (considering in each cluster only requests from D^L).

In the CL algorithm, we start with a fixed number of clusters K and partition V into K clusters. Clustering of C implies clustering of D. Let us denote $D = \{D_1, D_2, ...D_K\}$ as the resulting partition of D. Then $D_k^L = D_k \cap D^L$ and $D_k^U = D_k \cap D^U$. We estimate a choice model for each D_k^L and get the coefficient vector Θ_k for cluster k which also imply labels in D_k^U. The entire CL algorithm enumerates all possible K and the best performed one is chosen. To guarantee the computation of "soft labels," it is required that the number of labeled choice set in the k^{th} cluster is greater than the minimum number of requests to estimate the model. For details of the CL algorithm, we refer to Zhu and Goldberg (2009).

D. XCL1 algorithm

While EM and CL are standard algorithms in SSL that we have adapted to discrete choice, the remaining two algorithms are derivations of CL that to the best of our knowledge have not yet been observed in the past (as such they are applicable in the context of general SSL.) Distinct from the CL algorithm, the XCL1 algorithm requires no number of clusters as an input. It develops the clusters automatically without a target number of clusters. A diagram representing the clustering process is displayed in Figure 2. In step (a), we partition V into two clusters C_1 and C_2 and obtain two centroids (the crosses in Figure 2). In step (b), the algorithm develops two random vectors passing through C_1 and $C_2's$ centroids with the length of each vector being equal to the average within cluster distance. We use the end points as the new centroids to further partition clusters C_1 and C_2 into two. In step (c) with BIC defined appropriately, it is assumed that since $bic(D_1, D_2, \Theta_1, \Theta_2)$ in step (c) is greater than $bic(D_1, \Theta_1)$ in step (b), and $bic(D_3, D_4, \Theta_3, \Theta_4)$ in step (c) is greater than $bic(D_2, \Theta_2)$ in step (b), we have four new clusters developed from the original C_1 and C_2. Otherwise, if the new BIC is less than the old one, then the old cluster would not be partitioned into two. The algorithm then repeats steps (b) and (c) to partition each of the new clusters until no more partitions can be developed. If at any point the number of labeled requests with a cluster is less than parameter m, we discard such a cluster (i.e. merge back to the parent cluster).

BIC of K clusters in the semi-supervised problem is defined as:

$$bic(D_1, ..., D_K, \Theta_1, ..., \Theta_K) =$$

$$\sum_{k=1}^{K} \log(P(D_k^{U^*}|\Theta_k)) - \frac{KR}{2} \log(\sum_{k=1}^{K} |D_k|)$$

Here R is the number of coefficients in the choice model (each cluster has the same coefficients) and $|D_k|$ the number of requests in cluster k. We refer to Algorithm 1 for details.

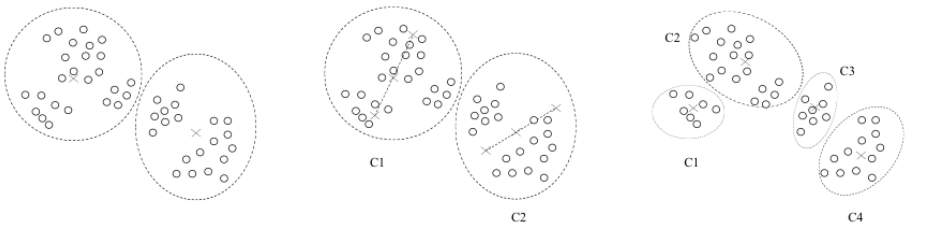
In the very first iteration, the computation of BIC starts with a model estimation based on D_k^L and then "soft labels" are assigned to D_k^U according to the coefficients of the model. After that, we combine D_k^L and $D_k^{U^*}$ together to re-estimate the model and get the BIC.

E. XCL2 algorithm

Instead of developing each cluster independently, XCL2 aims at partitioning a pair of clusters into three clusters starting from one more potential centroid. In Figure 3, step (a) is the starting point with 2 clusters which can be obtained as in Algorithm 1 in an iteration. Once Algorithm 1 terminates as it can no longer split clusters, we start a new partitioning direction. In the example below, suppose Algorithm 1 does not split the two clusters. Then we invoke steps (b) and (c) in Figure 3. Step (b) first randomly selects two clusters and constructs a connecting vector (the dotted line) between the two clusters' centroids (e.g. centroid 1 and centroid 2). Then it computes a new centroid (e.g. centroid 3) which lies in the middle of the connecting vector. In step (c), the algorithm partitions the data by 3-means into three clusters based on the three initial centroids (centroid 1, centroid 2 and centroid 3) if and only if $bic(D_1, D_2, \Theta_1, \Theta_2)$ in step (b) is less than $bic(D_1, D_2, D_3, \Theta_1, \Theta_2, \Theta_3)$ in step (c). If the algorithm can successfully split the initial two clusters into three, it then invokes Algorithm 1, followed by steps (b) and (c) to further split into three clusters until no more partitions can be developed. Details of the XCL2 algorithm are presented in Algorithm 2.

III. Computational Studies

In order to evaluate the performance of the algorithms, we used a labeled hotel data set and removed the labels from a subset of the data. We applied the ROL model in this case. The reason we apply the ROL model is that it proliferates the number of choice sets in our hotel data set, and as a benchmark



(a) Initial 2 clusters

(b) Cluster development: randomly generate a vector going through centroids

(c) Cluster development: partition 2 clusters into 4 clusters

Fig. 2: Illustration of XCL1 algorithm

Algorithm 1: XCL1

Input: D^L, D^U; output: Θ
Initialize: $K = 2$
K_{max} //Maximum number of clusters
$R_C = \{C_1, C_2\}$ //By partitioning $\{X^L \cup X^U\}$ based on V
$IterNumMax$ //Maximum number of iterations for cluster split
while $K < K_{max}$ **do**
  $R = \emptyset$ //New clusters
  foreach $C_k \in R_C$ **do**
    $IterNum \leftarrow 0$
    while $IterNum < IterNumMax$ **do**
      Generate a vector g //Based on $uniform(-1, 1)$
      $\hat{g} = \frac{g}{||g||}$
      $\overline{u_k} = \frac{1}{|C_K|} \sum_{v_i \in C_k} ||v_i - c^k||$
      $c_{new} = \{c^k + \overline{u_k}\hat{g}, c^k - \overline{u_k}\hat{g}\}$
      Partition C_k into C_{k1} and C_{k2} //Use 2-means with c_{new} as starting centroids
      if $l(C_{k1}) > m$ *and* $l(C_{k2}) > m$ **then**
        Break
      $IterNum$++
    end
    if $l(C_{k1}) > m$ *and* $l(C_{k2}) > m$ **then**
      Compute Θ_{k1}, Θ_{k2} for D_{k1}, D_{k2}
      if $bic(D_{k1}, D_{k2}, \Theta_{k1}, \Theta_{k2}) < bic(D_k, \Theta_k)$ **then**
        $R = R \cup \{C_k, (C_{k1}, C_{k2})\}$
  end
  if $R = \emptyset$ **then**
    Break
  else
    Update R_C using R
  end
end
Compute Θ based on R_C

study the hotel data set is relatively small compared to the airline data set presented later. As the ROL model can be transformed into a series of MNL models, the formulations and algorithms in Section 2 hold. In the second case study, we analyzed an airline data set by manually labeling a small subset with a rule. The proposed algorithms were then applied in this case. Prediction accuracies are provided.

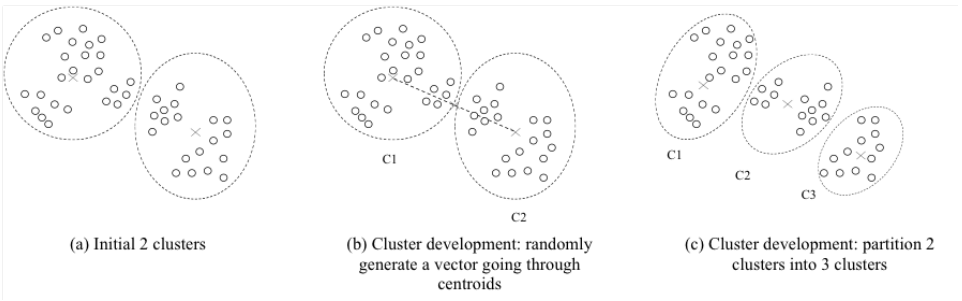
A. Hotel case study

We start with the hotel study where all of the choices are available. A data description and experimental design, performance analyses and algorithm recommendation are presented below.

1) Data description and experimental design: Data was collected from five U.S. properties of a major hotel chain for check-in dates between March 12th, 2007, and April 15th, 2007 (Bodea et al., 2009) and made publicly available. We refer to Bodea et al. (2009) for details about the data format and descriptive statistics. In total, there are 3,140 requests. Since each subsidiary of this hotel chain has different names for a similar service, we aligned the service names based on the descriptions of each hotel's service packages. For example, "accommodation combined with in city activities" and "accommodation combined with special event activities" are aligned with the name "accommodation with activities." On the other hand, some hotels have specific descriptions of room type, such as "non-smoking queen beds room" or "smoking queen beds room." We simply grouped them based on their upper level description which is their room type (e.g. queen beds room). For details about feature descriptions, see Tables 1 and 2 in the appendix.

The data is set to estimate classic choice models such as the MNL models instead of the ranked choice models. So we first transformed this data set into a ranked data set. To convert, we first estimate the MNL model's coefficients and use them to re-estimate each choice utility and the probability of choosing each one. Then we rank the choices based on their probabilities in a decreasing order. If there is a tie, we randomly rank them. These steps have been similarly applied in Lhéritier et al. (2018). With the ranked data, we applied the ROL model instead of the traditional MNL model. To evaluate the performance of the predicted rank, four metrics were applied: log-likelihood for ranked data, rank difference (e.g. predicted rank against the true rank), position difference and reciprocal rank. The descriptions are summarized below.

Rank-ordered log-likelihood (ROLIK). In *ROLIK*, we roll up the log-likelihood of all choices within each choice set while in MNL we only consider the chosen choice. Denote r_i as



(a) Initial 2 clusters

(b) Cluster development: randomly generate a vector going through centroids

(c) Cluster development: partition 2 clusters into 3 clusters

Fig. 3: Illustration of XCL2 algorithm

Algorithm 2: XCL2

Input: D^L, D^U; output: Θ
Initialize: $K = 2$
K_{max} //Maximum number of clusters
$R_C = \{C_1, C_2\}$ //By partitioning $\{X^L \cup X^U\}$ based on V
CP //A set containing all pairs of clusters from R_C
while $K < K_{max}$ **do**
  Execute Algorithm 1
  if R_C *has not been updated* **then**
    $R = \emptyset$
    foreach $\{C_{k1}, C_{k2}\} \in CP$ **do**
      $c^{k3} = \frac{1}{2}(c^{k2} + c^{k1})$
      Partition $C_{k1} \cup C_{k2}$ into $\{C_p, C_q, C_o\}$ //Use 3-means with $\{c_{k1}, c_{k2}, c_{k3}\}$ as starting centriods
      Compute $\Theta_p, \Theta_q, \Theta_o$ for D_p, D_q, D_o
      if $l(C_p) > m$, $l(C_q) > m$, $l(C_o) > m$, $bic(D_p, D_q, D_o, \Theta_p, \Theta_q, \Theta_o) < bic(D_{k1}, D_{k2}, \Theta_{k1}, \Theta_{k2})$ **then**
        $R = R \cup \{(C_{k1}, C_{k2}), (C_p, C_q, C_o)\}$
        Break //We break once a better partition is found to save computation time
    end
  if $R = \emptyset$ **then**
    Break
  else
    Update R_C using R
  end
end
Compute Θ based on R_C

the rank of choices in X_i. According to Fok et al. (2007), the log-likelihood of rank r_i in choice set X_i is:

$$\log(P(r_i|\Theta)) = \log(P(U_{i1} > U_{i2} > ... > U_{i|X_i|})) = \log(\prod_{j=1}^{|X_i|-1} \frac{\exp(U_{ij})}{\sum_{j=i}^{I} \exp(U_{ij})})$$

A better model yields a higher *ROLIK* value.

Rank difference (RD). Kendall's τ is a non-parametric estimator of the correlation between two vectors of random variables. Let $\tau(x, y)$ be the Kendall's τ between rank x and y. Denote r^T, r^E and r^R as the true rank, the estimated rank and a randomly generated rank, respectively. Since Kendall's τ is between $[-1, 1]$, we capture $\frac{1-\tau(r^T, r^E)}{2}$ and $\frac{1-\tau(r^T, r^R)}{2}$ which are within $[0, 1]$. For this metric, a lower value indicates that the two ranks are similar.

Position difference (PD). Given choice set X_i, denote $j_{(i)}$ as the index in the original data set with $\delta_{ij_{(i)}} = 1$. Let r_i^E be the estimated rank and p be the estimated rank of choice $j_{(i)}$ in r_i^E. We define *PD* as $\frac{p-1}{|X_i|-1}$. Using this metric, a better model yields a lower value.

Reciprocal rank (RR). The *mean reciprocal rank (MRR)* for a set of Q queries is defined as $MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{p_i}$ where p_i is the rank of the first correct response in every i in returned responses. In our case, p_i is the rank of the chosen choice in the original data set. Using this metric, a better model generates a higher value.

The goal of the experiment is to evaluate each algorithm's prediction accuracy under different labeled percentages. So the experimental design includes two parts: (1) transforming the original data set into a ranked data set; (2) designing the 10-fold cross validation experiment and the baseline. We use Label-q% to indicate the experiment with q% labeled data and (100-q)% unlabeled data. We transformed the original data set since this allowed us to apply the ROL model. We used cross validation since it provided a balanced evaluation of algorithms' prediction and reduced variability. With high q%, the marginal benefits of applying SSL may be reduced so we focus on Label-10% to Label-60% cases in increments of 10%. In each cross validation step we want the ranks of labeled requests to be consistent, for example, Label-20% uses labeled data from Label-10% which should have consistent ranks in requests appearing in both. In order to achieve this property, we proceed as follows. We have first evenly divided the data set into 10 batches (see Figure 4) so each batch represents 10% of data. We have then randomly selected 4 batches and removed their labels as they would be unlabeled throughout the experiment. The next step is to train an MNL model on each of the remaining 6 batches independently. We use each batch's coefficients to compute each choice's utility and rank them within each request. If we want to conduct the experiment Label-20%, we draw the ranked batches (i.e. the first and second batches) and remove the rank labels in the remaining 4 batches (see Figure 5). In Figure 5, (R) represents that choices in batch are ranked.

To measure the prediction accuracy, we first explain the use of train and test sets. Since we want the test set to be consistent

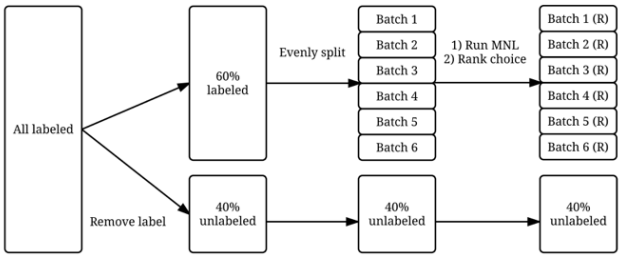


Fig. 4: Experimental design: generation of ranked data

across all experiments, we always select it from batch 1 (which is present in all experiments). In a single fold experiment, we split each batch into 10% and 90%. All 90% sets are used for training (in addition to all unlabeled data) and the 10% from batch 1 is used as test. This is repeated 10 times over the 10 different folds. Based on such partitions, each testing set in one fold of cross validation has approximately 30 observations and each observation has around 20 choices to rank which is still complex to predict. The baseline model for an experiment at Label-$q\%$ consists of the ranked logit model estimated on all labeled train instances (i.e. $q\% \times 90\%$).

The model estimation part has been coded in Spark Scala (Zaharia et al., 2010) using the stochastic gradient descent (SGD) method due to its size. Due to this choice of the algorithm, statistical significance values are not applicable. The cluster includes four 2.2GHz Xeon CPUs with each of them having 8 cores and 32 GB memory. Cloudera version CDH 5.8 is the underlying Hadoop distribution (Shvachko et al., 2010). After parameter tuning for all combinations of step size (from 1 to 40 with increment of 1) and sampling rate (from 0.2 to 0.8 with increment of 0.2), we found step size=40 and sampling rate=0.2 yielded the highest log-likelihood. Spark uses the learning rate of $\frac{Step\ size}{\sqrt{Number\ of\ iterations}}$. So we used this setting for the hotel case.

In the experiments, the EM algorithm stops when the coefficient change $||\frac{1}{t+1}\sum_1^{t+1}\Theta^{t+1} - \frac{1}{t}\sum_1^{t}\Theta^{t}|| < 0.01$. In the EM algorithm, due to the computational cost of the 10-fold cross validation steps and model estimation (the EM algorithm converges slowly), we randomly draw β percent unlabeled choice sets, and assign them the "soft-labels" in each iteration and ran ranked logit model on the current labeled data and random draw. To implement the CL and XCL algorithms, we use the Lloyd algorithm in the clustering process as it can assign starting centroids. K-means and Euclidean distance are used in the clustering process.

2) Performance analyses: In this performance analyses section, Figures 6 and 7, we compare the four metrics computed from the proposed algorithms to the baseline prediction. All metric values are compared to the baseline and numbers are presented in percentage. For example, in the Figure 6's upper chart, the y-axis represents $\frac{ROLIK\text{-}Baseline}{Baseline} \times 100\%$. For the CL algorithm, we experimented with the different number of clusters (from 2 to 6 with increments of 1) and report only the best one. Similarly, in the EM algorithm, we experimented with two β percent requests (i.e. 5% and 10%) and report only the best one in the figure. With regard to Kendall's τ against a random rank, we observed (results not shown here) that the values for all experiments were around 0.5 which means $\tau(r^T, r^R)$ tended to be zero. It indicates that the results are distant from random decisions. The out-of-sample results of the baseline models are presented in Appendix Table 3 and the main discoveries are summarized in Table 4. Table 3 shows that when q increases, the baseline model's performance deteriorates. In this experiment, each baseline model is tested on the same test set. Additional labeled data may add more noise to the model calibration process; with more labeled data, the trained coefficients do not easily capture the regression correctly (i.e. R^2 is lower).

For the metric *PD* and *RR* (see Figure 7), the trend varies. The pattern in different SSL-DCM algorithms is not consistent with what is observed in log-likelihood and rank difference. For example, the algorithms generally perform better in Label-40% than in Label-50% but are worse than baseline in other labeled percentage for metric *PD*. A possible reason accounting for the inconsistent pattern is that this metric measures the position of the originally chosen choice in the estimated rank, however the ROL model is to maximize the log-likelihood of all ranked choices. We also categorize some phenomena and insights in Table 6.

3) Algorithm recommendation: To provide recommendations for using the algorithms, we first analyze their computational times which are displayed in Figure 8. The figure shows the combined running time for each algorithm (e.g. for the CL algorithm, we add the running time of all possible clusters together). In Figure 8, when labeled percentage is below 20%, the EM algorithm is not the most time consuming method, yet when $q\% \geq 20\%$ its running time grows quickly. This indicates the EM algorithm may not fit larger $q\%$. Meanwhile, CL spends similar time compared to XCL1. When $q\%$ is small (e.g. 10% or 20%), the time difference between XCL1 and XCL2 is larger than a higher $q\%$ (e.g. 30%-60%).

Based on the previous prediction accuracy and computational time displayed in Figure 8, in Figure 9 we provide recommendations for using the algorithms under different scenarios. The x-axis represents the computation time while the y-axis represents the difference between algorithms' prediction and the baseline by using the rank difference metric. We use this metric since it considers the relations among choices in a choice set. For x-axis and y-axis, a lower value indicates a better algorithm. If the data set contains 10% labeled data, we recommend to use CL if there is time sensitivity and XCL1 if higher prediction accuracy is required. If the data set contains 30% or 50% labeled data, we recommend to use XCL1 if there is time sensitivity and XCL2 if high prediction accuracy is required. Based on this, we discover that the XCL1 algorithm's prediction accuracy is better and more consistent when labeled data increases. Label percentages 20%, 40%, 60% are not displayed in Figure 9 but we observe that XCL1 always lies in the efficient boundary (e.g. the dotted lines in Figure 9), and for 40% and 60% it yields best prediction accuracy.

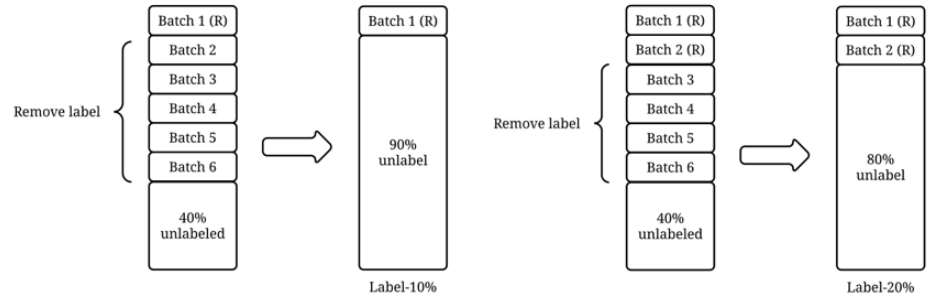


Fig. 5: Experimental design: Label-10% and Label-20% cases

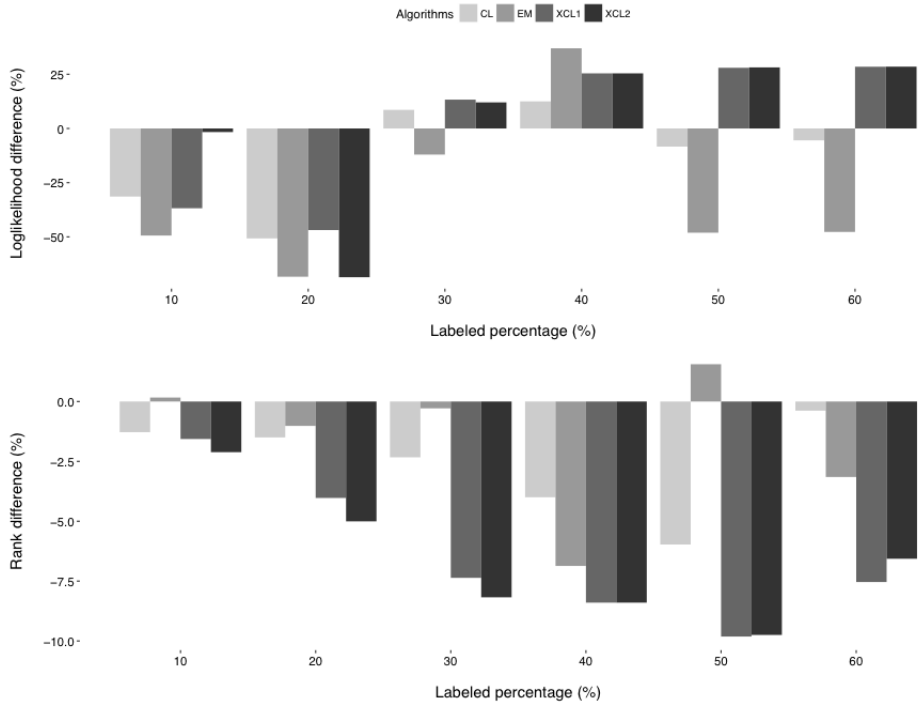


Fig. 6: Prediction accuracy (difference to the baseline) plots for log-likelihood and rank difference

B. Airline shopping case study

1) Data introduction: All of the demand and itinerary data were from a GDS provider. The case is based on a single international market with travelers' departure dates spanning one year. Since the entire data set is large, we only focused on round-trips. We use the term "service" for one or multiple flight legs connecting travelers' origin and destination. So one round-trip includes two services. Since we were not provided the actual bookings made by travelers, we came up with a heuristic method to infer some of the bookings. We have 11,000 requests and almost 2 million itineraries returned by the GDS. According to a survey conducted by Ipsos Public Affairs (2016), "Total travel price" was ranked as important by 86% of those respondents who flew in 2015 and the next three most cited factors were airline schedule (83%) and total travel time (77%). Inspired by this fact, we made a modest assumption that a traveler purchases the itinerary with the lowest fare and the best match to his or her travel preferences. To measure the best match of travelers' preference, we first define quality H_{ij} of itinerary j for request i.

$$H_{ij} = |t_i^{d1} - t_j^{d1}| + |t_i^{d2} - t_j^{d2}| + |t_i^{a1} - t_j^{a1}| + |t_i^{a2} - t_j^{a2}| + |e_i - e_j|$$

t_i^{d1}, t_j^{d1}−Requested departure time of the first service and itinerary's departure time of the first service

t_i^{d2}, t_j^{d2}−Requested departure time of the second service and itinerary's departure time of the second service

t_i^{a1}, t_j^{a1}−Requested arrival time of the first service and itinerary's arrival time of the first service

t_i^{a2}, t_j^{a2}−Requested arrival time of the second service and itinerary arrival time of the second service

e_i, e_j−Elapsed time of request i and itinerary j, respectively.

A lower H_{ij} yields a better quality of response. Let F_{ij} be the fare of itinerary j in response to request i and let LF_i be the lowest fare in request i. Similar to LF_i, LH_i is defined as the lowest H_{ij}. For δ_{ij}, we set $\delta_{ij} = 1$ if $H_{ij} = LH_i$ and $F_{ij} = LF_i$ and $\delta_{ij} = 0$, otherwise. In most records,

Table 4. Experiment analysis-1

Metric	Phenomena	Insights
ROLIK	When labeled percentage is low ($q\%<30\%$), all algorithms yield negative values.	(1) When using ROLIK as the criterion, the SSL-DCM algorithms perform better in low labeled percentage scenarios. (2) The algorithms work best at 20%.
ROLIK	When labeled percentage is high ($q\%\geq30\%$), algorithms yield more positive values.	Using *ROLIK*, it indicates that the algorithms are not stable in finding relative utility differences between choices when labeled percentage is high. In one case when SSL and baseline both predict rank correctly, SSL's *ROLIK* may not be higher than the baseline since the probability depends on utility differences. In another case, even though SSL predicts a better rank than the baseline, it is still not guaranteed its *ROLIK* is higher. A simple example is presented in Table 5 showing that even when the semi-supervised prediction can predict the correct ranking ($U_3 > U_2 > U_1$) and the baseline predicts incorrectly ($U_2 > U_3 > U_1$), its *ROLIK* measurement may not be higher than the baseline.
ROLIK	When labeled percentage is high ($q\%\geq30\%$), the EM algorithm generates negative results.	Using *ROLIK*, the EM algorithm is more stable with respect to the labeled percentage. This is reasonable since Bayesian approaches rely on distribution fitting while clustering based algorithms are more about distance minimization.
ROLIK	Except for ($q\%=40\%$), the EM algorithm generates the lowest values.	Using *ROLIK*, the EM algorithm outperforms other algorithms in most situations.
RD	10% to 50%: $q\% \uparrow$ implies *RD* $\downarrow$ and on average *RD*<0; 50% to 60%: $q\% \uparrow$ implies *RD* $\uparrow$ and on average *RD*<0	(1) When using *RD* as the predicting criterion, the algorithms have excellent performance as almost all results are better than the baseline. (2) In general, the algorithms have a much better prediction power when $q\%$ is below a certain level (e.g. 50%).
RD	Clustering based algorithms (i.e. CL, XCL1 and XCL2) are always below zero while the EM algorithm is above zero at 50%.	When using *RD* as the predicting criterion, the clustering based algorithms work better and are more stable than the EM algorithm.
RD	XCL2 is lower than XCL1 before 40% but higher than XCL1 after 40%.	When using *RD* as the predicting criterion, XCL2 is better in low labeled scenario. But when labeled data grows, the power diminishes.

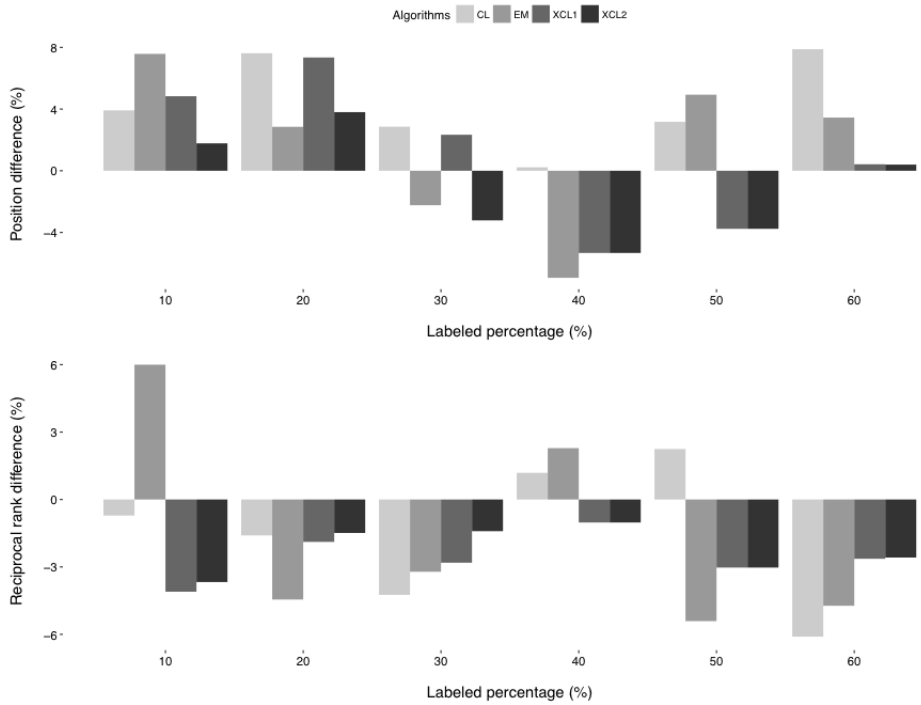


Fig. 7: Prediction accuracy (difference to the baseline) plots for position difference and reciprocal rank

arrival times are missing. We only use H_{ij} as a heuristic way to measure the distance between one itinerary and the ideal choice. If a request states both departure and arrival times, then the heuristic is more precise. In the choice model, we discard arrival times as features and only include departure times because all requests in the train and test datasets have values.

The assumption is that if there is an itinerary with lowest fare and lowest deviation from preference, then it is booked. If $\sum_{j=1}^{|X_i|} \delta_{ij} = 1$, then we assume request is labeled. If not, the request is unlabeled. After finding the labels using the strategy described above, we split the initial data set into 1,000 labeled

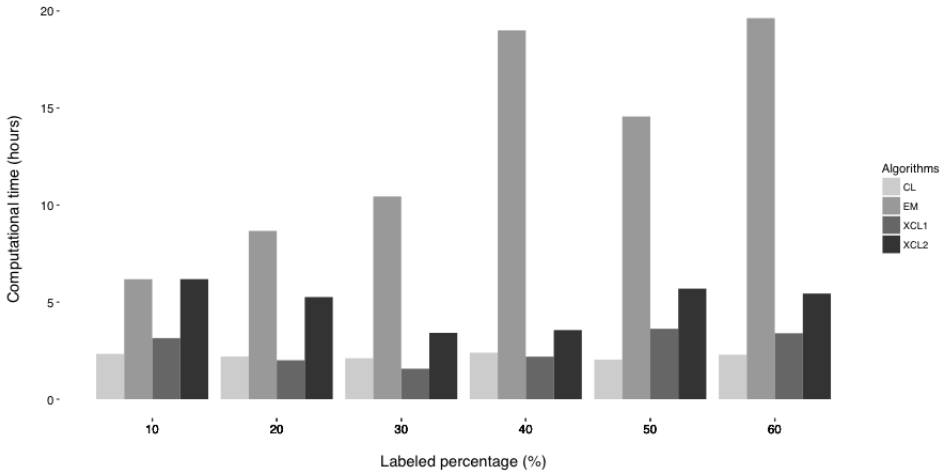


Fig. 8: Combined computation times for SSL-DCM algorithms

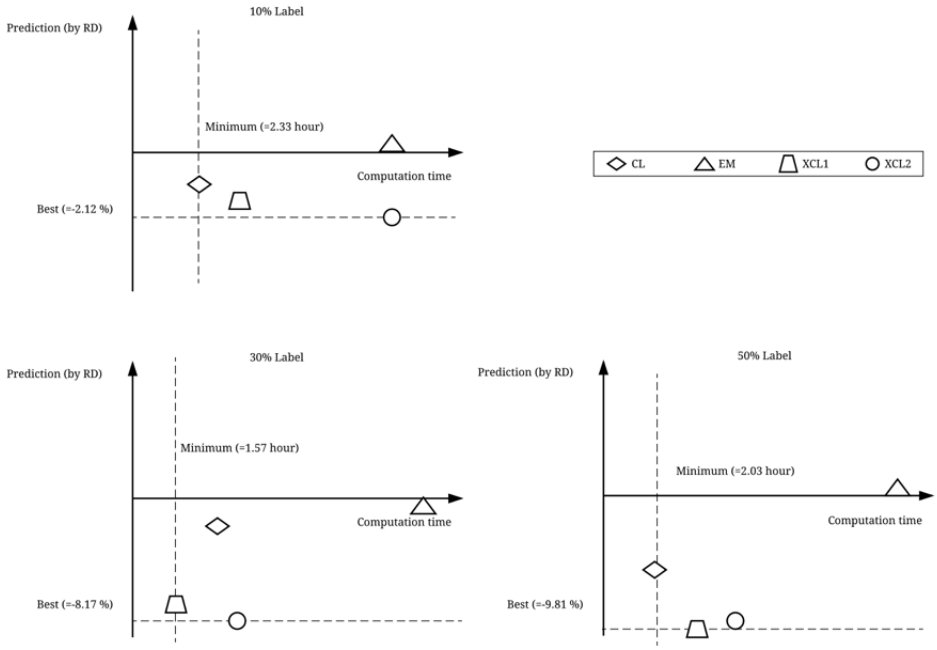


Fig. 9: Algorithm recommendations for different scenarios

Table 5. An example displaying *ROLIK* prediction

True rank	Baseline utility	SSL utility
Choice 3	$U_3 = 4$	$U_3 = 5$
Choice 2	$U_2 = 5$	$U_2 = 3$
Choice 1	$U_1 = 1$	$U_1 = 2$
ROLIK	$\log(\frac{4}{10} \times \frac{5}{6}) \approx \log(0.33)$	$\log(\frac{5}{10} \times \frac{3}{5}) = \log(0.3)$

requests and 10,000 unlabeled requests. Then we use this data set for the following computational analyses.

2) Computational analyses: With the airline data, we know travelers' underlying preferences for airlines. For example, a request may state that legacy carriers are preferred. But in the experiment, we left them out from the quality formula since it is a categorical feature and unrelated to time. In the logit model, we consider features shown in Appendix Table 8 related to the calculation of H_{ij} and F_{ij}. In order to verify the prediction accuracy, we expect the yielded model coefficients would select an itinerary with low fare and good quality of response. Besides the intercept, the model considers three alternative specific features. The first feature f_{ij}^1 captures the gap between each itinerary j's fare and the average fare in X_i, formally $f_{ij}^1 = F_{ij} - \frac{1}{|X_i|}\sum_{j=1}^{|X_i|} F_{ij}$. We apply this feature instead of using the fare directly because the fare range of different requests may vary a lot, even in the same market, and different advanced purchase days may contribute to distinct fare levels. The second and third features are defined as $f_{ij}^2 = t_i^{d1} - t_j^{d1}$ and $f_{ij}^3 = t_i^{d2} - t_j^{d2}$, respectively. Ideally we should include the calculation of requested elapsed time and requested arrival time in the calculation of H_{ij}, but both fields are empty in most of the records, and they were left

Table 6. Experiment analysis-2

Metric	Phenomena	Insights
PD	XCL2 is lower than XCL1 before 40% but similar to XCL1 after 40%.	Similar to *RD*, the XCL2 algorithm works better than XCL1 only when $q\%$ is less than a certain value.
PD	XCL2 always generates the lowest or second lowest values.	Using *PD*, the XCL2 algorithm works better than other algorithms.
RR	XCL2 is higher than XCL1 before 40% but similar to XCL1 after 40%.	Same as above.

out. In addition, we did not compute the absolute value for f_{ij}^2 and f_{ij}^3 because compared to a traveler's desired time, an earlier departure is different from a later departure. On the other hand, in the formula of H_{ij} we consider the absolute values of the differences since we do not want the negative values offset the positive values.

Similar to the hotel case, we used the SGD method in Scala since the logit model estimation was computationally expensive and it took more than 120 hours in R to estimate the model with the entire data set. The input data was stored in HDFS and the cluster environment was the same as in the hotel case. We also tuned the SGD parameters by trying combinations of step sizes and sampling rates and found step size=7 and sampling rate=0.8 yielded the highest log-likelihood. So we use this setting for the airline case.

Due to confidentiality, we do not present the exact coefficients of SSL-DCM here. Since we have coefficients for fare and also time related features, we can estimate the value of time (i.e. dollars per hour). The coefficients of fare gap and departure time have units $utility/dollar$ and $utility/hour$, respectively. So we simply compute the ratio of these two and get the value of time. We discover that travelers value more the first departure time than the second departure time. For example, the EM algorithm indicates that the value of the first departure time is between \$250-\$300 per hour while the value of the second departure time is between \$150-\$200 per hour.

We designed two metrics to evaluate the prediction accuracies of the coefficients. As mentioned before due to the strategy of labeling the 1,000 requests (i.e. if there is an itinerary with lowest fare and lowest deviation from preference, then it is booked), we expect the coefficients to predict an itinerary with the lowest fare and the best match. Let us first define the term "lowest $p\%$ records." In each X_i, we sort the itineraries by F_{ij} in ascending order and find $p\%$ itineraries with lowest fare, denoted as A_i. In a similar way, we sort the itineraries by H_{ij} for each X_i and find $p\%$ itineraries with lowest H_{ij}, denoted as B_i. Let Z_i be $A_i \cup B_i$. Suppose F_i^p is the fare and H_i^p is the quality of the predicted itinerary of request i. Then for each request i, we define the prediction accuracies as

$$MF_i = \frac{F_i^p - \frac{1}{|Z_i|}\sum_{j=1}^{|Z_i|} F_{ij}}{\frac{1}{|X_i|}\sum_{j=1}^{|X_i|} F_{ij}}$$

and

$$MH_i = \frac{H_i^p - \frac{1}{|Z_i|}\sum_{j=1}^{|Z_i|} H_{ij}}{\frac{1}{|X_i|}\sum_{j=1}^{|X_i|} H_{ij}}.$$

For these metrics, a lower value indicates a better prediction.

We tested the metrics with $p\%$ ranging from 10% to 50% with the step size of 10%. The results are presented in Table 7. From Table 7, we observe that the generated coefficients have a good prediction in terms of fare. For example, both the CL and XCL1 or XCL2 algorithms are better than the average when $p\%\geq 10\%$. However, with respect to response quality, only the clustering based algorithms are slightly close to the benchmark and this requires $p\%$=50%. One explanation for this phenomena is that f_{ij}^2 and f_{ij}^3 in the H_{ij} formula are captured by their absolute values which could lead to some discrepancies in terms of prediction accuracy.

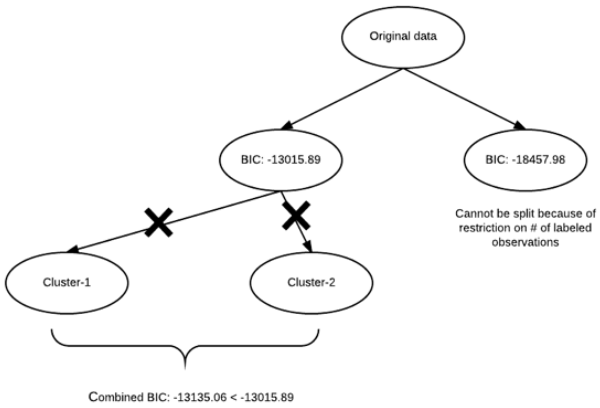


Fig. 10: Clustering development in XCL1

We also compared our semi-supervised learning algorithms to a supervised learning model. The baseline in Table 7 uses a choice model estimation only on labeled data. The estimation uses the same SGD parameters (step size and sampling rate) mentioned before. In the comparison, the XCL1 model outperforms the baseline at every $p\%$. In addition, the CL model is better than the baseline in terms of MH_i. However, the EM algorithm is not favored in this case. This result reinforces the assumption that relying on labeled data alone may lead to bias and the clustering based model may better prevent the bias.

To better understand the XCL algorithms, we present the clustering development tree of the XCL1 algorithm (see Figure 10). The algorithm first partitions the original data into two clusters. Then each cluster is not able to be further partitioned. For one branch it is because the combined BIC of the potential split is lower than its parent node. For another branch, the algorithm cannot find enough number of observations in each cluster to guarantee the model estimation.

Figure 11 shows how the XCL2 develops new clusters based on the initial two clusters from XCL1. First of all, XCL2 split the two clusters into three by initializing a new center

Table 7. Prediction accuracies

		10%	20%	30%	40%	50%
MF_i (in %)	CL	1.71	**-1.68**	**-4.08**	**-5.98**	**-7.66**
	EM	29.70	24.81	21.79	19.28	17.13
	XCL1	**-3.91**	**-7.10**	**-9.26**	**-11.08**	**-12.72**
	XCL2	2.18	**-1.49**	**-3.75**	**-5.69**	**-7.44**
	Baseline	**-0.79**	**-4.17**	**-6.52**	**-8.36**	**-9.99**
MH_i (in %)	CL	35.47	26.25	15.35	8.49	3.02
	EM	56.62	45.80	32.81	24.36	17.94
	XCL1	36.11	26.13	15.99	9.76	4.42
	XCL2	37.66	27.12	16.63	10.18	4.65
	Baseline	36.90	27.50	17.03	10.33	4.83

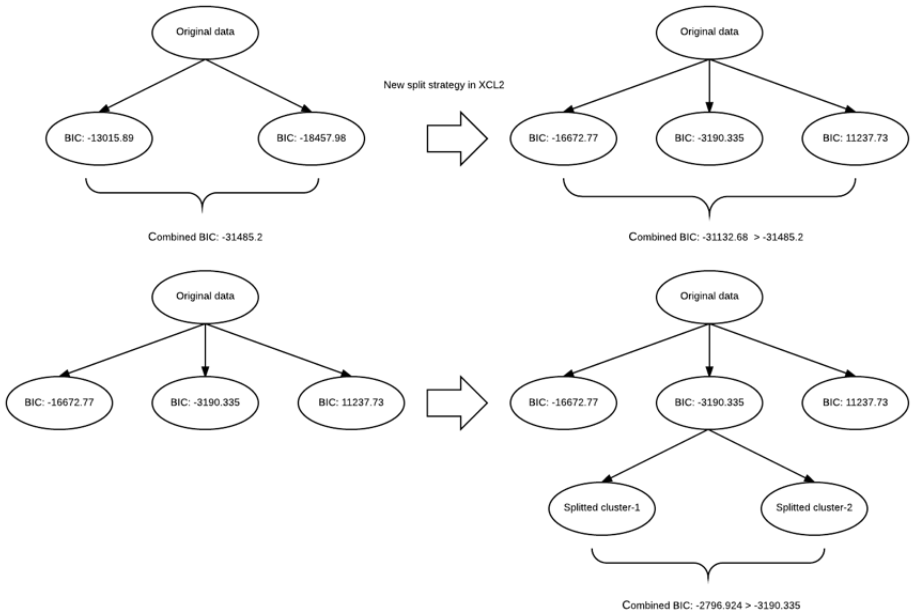


Fig. 11: Clustering development in XCL2

lying between original centers as the combined BIC for three clusters in higher than the combined BIC for the original two clusters. Then it further split the middle cluster into two because the BIC criterion holds. Later, the algorithm stopped since no more splits could be found.

IV. Conclusion

Innovative transportation modes (i.e. ride sharing, bicycle sharing) and more emphasis on customer personalization in more traditional travel industry (i.e. hotel and airline) bring lots of new data and novel problems. The rise of various machine learning methods and platforms also provide to transportation planners opportunities to combine traditional econometric models with machine learning algorithms. Inspired by this trend, the SSL-DCM algorithms discussed in this paper explore new approaches to estimate discrete choice models. The new approaches provide robust computational capability in solving large-scale transportation problems. The embedded algorithms can solve problems with limited labeled requests and many unlabeled requests. Besides adapting classic SSL algorithms such as EM and CL, we design two new algorithms XCL1 and XCL2 which can automatically segment the requests with an adapted BIC criterion. The SSL-DCM algorithms have been tested on two case studies. In the hotel case, we evaluated the prediction accuracies of the SSL-DCM algorithms using 10-fold cross validations. The main findings from this case study are summarized as follows.

- In general, the XCL1 algorithm has consistently good prediction accuracy and relatively low computation time.
- When the data set contains around 10% labeled data, we recommend to use CL if satisfactory solutions must be obtained quickly, and XCL1 if high prediction accuracy is needed.
- When the data set contains 20%, 30% or 50% labeled data, we recommend to use XCL1 if computational time is at a premium and prediction accuracy can be slightly sacrificed, and XCL2 if high prediction accuracy is needed.
- When the data set contains 40% or 60% labeled data, we recommend to use XCL1.

In the airline shopping case, we explored the travelers' purchasing behavior with a real data set from a GDS provider.

The main findings are as follows.

- Results from the EM algorithm shows that travelers' value of time for first departure time is between \$250-\$300 per hour while the value of second departure time is between \$150-\$200 per hour.
- With regard to prediction accuracy, XCL1 outperforms CL, EM and XCL2 in this case.
- XCL2 is no better than XCL1 in terms of prediction accuracy even if finds more possible partitions in this case.

Future studies include improving the current SSL-DCM algorithms to yield a lower computation time and also to test them in more complex discrete choice models such as nested logit or cross-nested logit models.

ACKNOWLEDGMENT

We are grateful to Sabre Holdings for providing the airline shopping data. Jie Yang has also been partially supported by a fellowship from the Transportation Center at Northwestern University.

REFERENCES

[1] B.R. 2015. No turning back the tide. The Economist. Retrieved on May 2nd, 2016: http://www.economist.com/blogs/gulliver/2015/06/travel-booking-sites.

[2] Basu, S., Banerjee, A., Mooney, R.J. 2002. Semi-supervised clustering by seeding. Proceedings of the 19^{th} International Conference on Machine Learning. 27-34.

[3] Bhat, C.R. 1997. Endogenous segmentation mode choice model with an application to intercity travel. Transportation Science. 31:34-48.

[4] Beggs, S., Cardell, S. and Hausman, J. 1981. Assessing the potential demand for electric cars. Journal of Econometrics. 16:1-19.

[5] Ben-Akiva, M.E. and Bierlaire, M. (1999) Discrete choice methods and their applications to short term travel decisions. Handbook of Transportation Science. Randolph W. Hall, ed.

[6] Bodea, T., Ferguson, M. and Garrow, L. 2009. Data set-choice-based revenue management: Data from a major hotel chain. Manufacturing and Service Operations Management. 11:356-361.

[7] Calfee, J., Winston, C. and Stempski R. 2001. Econometric issues in estimating consumer preferences from stated preference data: A case study of the value of automobile travel time. The Review of Economics and Statistics. 83:699-707.

[8] Carrier,E.2008. Modeling the choice of an airline itinerary and fare product using booking and seat availability data. Ph.D. thesis. Department of Civil and Environmental Engineering. Massachusetts Institute of Technology, Cambridge.

[9] Chapelle, O., Schlkopf, B. and Zien, A. 2006. Semi-supervised Learning. MIT Press, Cambridge.

[10] Dempster, A.P., Laird, N.M. and Rubin, D.B. 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B (Methodological). 39:1-38.

[11] Demiriz, A., Bennett, K. and Embrechts, M. 1999. Semi-supervised clustering using genetic algorithms. Proceedings of Artificial Neural Networks in Engineering. 809-814.

[12] Depaire, B., Wets, G. and Vanhoof, K. 2008. Traffic accident segmentation by means of latent class clustering. Accident Analysis and Prevention. 40:1257-1266.

[13] Dara, R., Kremer, S. and Stacey, D. 2002. Clustering unlabeled data with SOMs improves classification of labeled real-world data. Proceedings of the 2002 IEEE World Congress on Computational Intelligence. 3:2237-2242.

[14] Fok, D., Paap, R., Dijk, B.V. 2012. A rank-ordered logit model with unobserved heterogeneity in ranking capabilities. Journal of Applied Econometrics. 27:831-846.

[15] Grira, N., Crucianu, M., and Boujemaa, N. 2004. Unsupervised and semi-supervised clustering: A brief survey. The 7^{th} ACM SIGMM international workshop on multimedia information retrieval. 9-16.

[16] Heimlich, J.P. 2016. Status of air travel in the USA. Ipsos Public Affairs. Retrieved on May 2nd, 2016: url=http://airlines.org/wp-content/uploads/2016/04/2016Survey.pdf.

[17] Shvachko, K., Kuang, H.R., Radia, S. and Chansler, R. 2010. The Hadoop distributed file system, Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). 1-10.

[18] Jain, A. K. 2010. Data clustering: 50 years beyond K-means. Pattern Recognition Letters. 31:651-666.

[19] Koppelman, F.S. and Sethi, V. (2000) Closed form discrete choice models. Handbook of Transport Modeling. Pergamon Press, Oxford. 211-228.

[20] Lindsey, C., Frei, A., Mahmassani, H.S, Alibabai, H., Park, Y.W., Klabjan, D., Reed, M., Langheim, G. and Keating, T. 2013. Online pricing and capacity sourcing for third party logistics providers. Technical report.

[21] Lhéritier, A., Bocamazo, M., Delahaye, T. and Acuna-Agost, R. 2018. Airline itinerary choice modeling using machine learning. Journal of Choice Modelling. DOI: 10.1016/j.jocm.2018.02.002.

[22] Liu, T.C., Yang, Y., Huang, G.B., Yeo, Y.K. and Lin, Z.P. 2016. Driver distraction detection using semi-supervised machine learning. IEEE Transactions on Intelligent Transportation Systems. 17:1108-1120.

[23] Mottini, A. and Acuna-Agost, R. 2017. Deep choice model using pointer networks for airline itinerary prediction. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 1575-1583.

[24] Newman, J., Ferguson, M.E. and Garrow, L.A. 2012. Estimating discrete choice models with incomplete data. Transportation Research Board. 2302:130-137.

[25] Nigam, K., McCallum, A. K., Thrun, S. and Mitchell, T. 2000. Text classification from labeled and unlabeled documents using EM. Machine Learning. 39:103-134.

[26] McFadden, D. (1978) Modeling the choice of residential location. *Transportation Research Record*. 72-77.

[27] Molesworth, B.R.C. and Koo, T.T.R. 2016. The influence of attitude towards individuals' choice for a remotely piloted commercial flight: A latent class logit approach. Transportation Research Part C. 71:51-62.

[28] Mohamed, M.G., Saunier, N., Miranda-Moreno, L.F. and Ukkusuri, S.V. 2013. A clustering regression approach: a comprehensive injury severity analysis of pedestrian-vehicle crashes in New York, US and Montreal, Canada. Safety Science. 54:27-37.

[29] Pelleg, D. and Moore, A.W. 2000. X-means: Extending K-means with efficient estimation of the number of clusters. The International Conference on Machine Learning. 727-734.

[30] Peng, Y., Lu, B.L. and Wang, S.H. 2015. Enhanced low-rank representation via sparse manifold adaption for semi-supervised learning. Neural Networks. 65:1-17.

[31] Podgorski, K.V. and Kockelman, K.M. 2006. Public perceptions of toll roads: A survey of the Texas perspective. Transportation Research Part A: Policy and Practice. 40:888-902.

[32] Schwenker, F. and Trentin, E. 2014. Pattern classification and clustering: A review of partially supervised learning approaches. Pattern Recognition Letters. 37:4-14.

[33] Steinberg, D. and Cardell, N. 1992. Estimating logistic regression models when the dependent variable has no variance. Communications in Statistics: Theory and Methods. 2:423-450.

[34] Vlahogianni, E.T., Karlaftis, M.G. and Golias, J.C. 2008. Temporal evolution of short-term urban traffic flow: a nonlinear dynamics approach. Computer-aided Civil and Infrastructure Engineering. 23:536-548.

[35] Vulcano, G., Ryzin, G.V. and Chaar, W. 2010. Choice-based revenue management: An empirical study of estimation and optimization. Manufacturing and Service Operations Management, 12: 371-392.

[36] Yasmin, S., Eluru, N., Bhat, C.R. and Tray, R. 2014. A latent segmentation based generalized ordered logit model to examine factors influencing driver injury severity. Analytic Methods in Accident Research, 1: 23-38.

[37] Zhang, Y. and Xie, Y. 2008. Travel mode choice modeling with support vector machines. Transportation Research Record. 2076:141-150.

[38] Zhu, X.J. 2008. Semi-supervised learning literature survey. Computer Sciences Technical Report 1530, University of Wisconsin-Madison.

[39] Zhu, X.J. and Goldberg, A.B. 2009. Introduction to semi-supervised learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers

[40] Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S. and Stoica, I. 2010. Spark: cluster computing with working sets. In Proceeding HotCloud.

Jie Yang graduated with Ph.D in Transportation System Analysis and Planning from Northwestern University in 2018. His research interests include vehicle routing, discrete choice model and machine learning (e.g. semi-supervised learning, deep Gaussian processes, etc.). He now works as an Operations Research Scientist at Staples. Previously, he has led projects with Sears Holding, Sabre and Bell Labs as well as serving a major China's airport company and a private equity firm as a consultant.

Sergey Shebalov graduated with Ph.D. in mathematics from University of Illinois at Urbana-Champaign in 2004. After graduation he worked at United Airlines as an Operations Research Analyst. In 2007 Sergey joined Sabre where he now holds a position of Senior Director of Operations Research. In the current role Sergey leads a team of 60 professionals working on development and implementation of analytics and decision support systems for commercial airlines planning, distribution and operations. Sergey's area of interest include airline network planning, revenue management, retailing, crew scheduling, airport management optimization and operations recovery.

Diego Klabjan is a professor at Northwestern University, Department of Industrial Engineering and Management Sciences. He is also Founding Director, Master of Science in Analytics. After obtaining his doctorate from the School of Industrial and Systems Engineering of the Georgia Institute of Technology in 1999 in Algorithms, Combinatorics, and Optimization, in the same year he joined the University of Illinois at Urbana-Champaign. In 2007 he became an associate professor at Northwestern. His research is focused on machine learning, deep learning and analytics with concentration in finance, transportation, sport, and healthcare. Professor Klabjan has led projects with large companies such as Intel, Baxter, FedEx Express, General Motors, United Continental, and many others, and he is also assisting numerous start-ups with their analytics needs.

APPENDICES

Table 1. Description of hotel data set

Name	Description
Booking ID	ID of a booking
Product ID	ID of a product
Purchased product	Binary indicator taking 1 if the product is purchased, 0 otherwise
Booking date	From 1 to 7, representing Monday through Sunday
Check-in date	From 1 to 7, representing Monday through Sunday
Check-out date	From 1 to 7, representing Monday through Sunday
Advanced purchase days	Time difference between the booking and check-in date
Party size	Number of adults and children associated with the booking
Length of stay	Number of nights (e.g. two)
Number of rooms	Number of rooms booked (e.g. three)
Membership status	Status in rewards program (0-not a member, 1-basic, 2-elevated, 3-premium)
Room type	Double, King, Queen, Regular, Special, Standard and Suite
Rate code	Rate1: Advance purchase
	Rate2: Rack rate (Unrestricted rate)
	Rate3: Rack rate combined with additional hotel service
	Rate4: Accommodation with activities or awards
	Rate5: Discounted rate less restricted than advance purchase
Price gap	Difference between the arrival price of one room product and the average price of the available choice set

Table 2. Hotel case: feature list

Clustering features (s_{ij}^{ISF})	Choice model alternative specific features (part of s_{ij}^{MSF})	Choice model interactive features (here we use to indicate interaction) (part of s_{ij}^{MSF})
Booking day of week	Price gap	Membership $\sim$ price gap
Check-in day of week	Room type	Advance purchase days $\sim$ price gap
Check-out day of week	Rate code	Income level $\sim$ price gap
Advanced purchase days		Income level $\sim$ room type (suite or not)
Party size		Income level $\sim$ discount or not (based on rate code)
Length of stay		Length of stay $\sim$ room type (suite or not)
Number of rooms		Length of stay $\sim$ discount or not (based on rate code)
Membership		Party size $\sim$ room type (suite or not)
Membership		Party size $\sim$ discount or not (based on rate code)

Table 3. Baseline experiment results

q %	ROLIK	RD	PD	RR
10 %	-207.913	0.036	0.195	0.523
20 %	-252.286	0.060	0.202	0.523
30 %	-258.400	0.061	0.204	0.511
40 %	-271.742	0.068	0.208	0.505
50 %	-280.556	0.076	0.214	0.506
60 %	-315.031	0.088	0.217	0.503

Table 8. Airline case: choice model feature list and description

Features	Description
Fare gap	Difference between one itinerary's fare with the average in its choice set
First departure difference	Difference between one itinerary's first departure time and the individual's preferred first departure time
Second departure difference	Difference between one itinerary's second departure time and the individual's preferred second departure time
Length of stay $\sim$ fare gap	Interactive terms between the trip's length of stay and the fare gap; length of stay is bucketed into three levels: level 1 (0$\sim$4 days), level 2 (4$\sim$7 days) and level 3 (7+ days)
Day of week $\sim$ fare gap	Interactive terms between the trip's departure day of week and the fare gap; day of week is bucketed into three levels: early week (Monday, Tuesday and Wednesday), late week (Thursday and Friday) and weekend (Saturday and Sunday)
Advanced purchase days $\sim$ fare gap	Interactive terms between individual's advanced purchase days and the fare gap; advanced purchase days is bucketed into four levels: early (84 days ahead), medium (28$\sim$84 days), late (14$\sim$28 days), rush (0$\sim$14 days)
Cabin $\sim$ fare gap	Interactive terms between individual's preferred cabin with fare gap; based on fare class's seniority we categorized cabin into four levels: business, economy, discounted economy and no preference