Recommending Product Sizes to Customers

Vivek Sembium
Amazon Development Center India
viveksem@amazon.com

Rajeev Rastogi
Amazon Development Center India
rastogi@amazon.com

Atul Saroop
Amazon Development Center India
asaroop@amazon.com

Srujana Merugu*
srujana@gmail.com

ABSTRACT

We propose a novel latent factor model for recommending product size fits {`Small`, `Fit`, `Large`} to customers. Latent factors for customers and products in our model correspond to their physical true size, and are learnt from past product purchase and returns data. The outcome for a customer, product pair is predicted based on the difference between customer and product true sizes, and efficient algorithms are proposed for computing customer and product true size values that minimize two loss function variants. In experiments with Amazon shoe datasets, we show that our latent factor models incorporating personas, and leveraging return codes show a 17-21% AUC improvement compared to baselines. In an online A/B test, our algorithms show an improvement of 0.49% in percentage of Fit transactions over control.

CCS CONCEPTS

•Information systems →Collaborative filtering; Recommender systems;

KEYWORDS

Personalization, Hinge Loss, Recommendation, Latent Factors, Ordinal Loss

1 INTRODUCTION

Product categories such as shoes and apparel have huge sizing variations across brands and locales, which makes it difficult to pick the right size for a desired product. For example, the catalog size to physical size mapping convention for Reebok is: 6 = 15cm, 7 = 17cm, 8 = 21cm, while that for Nike is: 6 = 16cm, 7 = 18cm, 8 = 22cm. Poorly chosen fits and wasteful strategies on part of the customers (e.g. buying multiple sizes of a product and returning the rest) result in very high return rates. A large number of customers also shy away from purchasing online in these product categories due to the effort involved in choosing the right sized product. Hence, e-commerce companies such as Amazon have a huge interest in being able to automatically provide highly accurate sizing guidance about products to customers.

*Work done while author was at Amazon.

RecSys '17, August 27–31, 2017, Como, Italy.
 978-1-4503-4652-8/17/08.
DOI: http://dx.doi.org/10.1145/3109859.3109891

In the size recommendation problem, a customer implicitly provides the context of a desired product by viewing the detail page of a product and requires a recommendation for the appropriate size variant of the product. For example, the customer might be viewing the detail page of Nike Women's Tennis Classic shoe and needs to choose from 10 different size variants corresponding to sizes from 6 to 15. Thus, given the context of a desired product, our objective is to recommend the appropriate size variant for a customer.

The problem of recommending sizes to customers is challenging due to the following reasons:

- *Data sparsity.* Typically, a small fraction of customers and products account for the bulk of purchases. A majority of customers and products have very few purchases.
- *Cold start.* The environment is highly dynamic with new customers and products (that have no past purchases) for which size recommendation need to be generated.
- *Multiple personas.* Many users tend to shop for their families (e.g., spouses, children) and occasionally even purchase gifts for friends. Thus, a customer account may have multiple underlying personas with widely varying sizes. In order to make accurate size recommendations, it is important to separate out these personas and learn the individual sizes for each customer persona.

In this paper, we propose a novel latent factor model for determining how a product of certain size fits a customer, i.e., {`Small`, `Fit`, `Large`}. Latent factors for customers and products in our model correspond to their physical true sizes (e.g. for dimensions like length and width), and are learnt from past product purchase and returns data. The true (latent) sizes of customers and products are used to predict the fit outcome for each product purchased by a customer. Specifically, for each (customer, product) purchase transaction, our model outputs a score that is a (linear) function of the difference between customer and product true sizes, which is then used to predict the outcome. We present a construction that reduces a loss function with ordinal outcomes to a new loss function with binary outcomes. We define Hinge loss and Logistic loss variants, and propose efficient algorithms with linear time complexity for computing customer and product true size values that minimize the two loss variants.

We show how features of customers (e.g. demographic information such as age, gender, past return rates) and products (e.g. brand, catalog size, category) can be incorporated into our model to handle data sparsity and cold start scenarios involving (new) customers and products for whom we have very few or no purchase transactions. Finally, we extend our formulation to address

the case where each customer corresponds to multiple personas and propose a hierarchical clustering algorithm to discover the different underlying personas for each customer. In experiments with Amazon shoe datasets, we show that our latent factor models incorporating personas, and leveraging return codes show a 17-21% AUC improvement compared to baselines. Furthermore, our algorithms can learn customer and product true sizes from large datasets containing millions of purchase transactions in a few minutes.

The remainder of the paper is organized as follows. After surveying prior work on recommendations and latent factor models in Section 2, we formally define our size recommendation problem in Section 3. In Section 4, we reduce our ordinal regression problem to multiple binary classification problems, and consider different binary loss function variants such as Hinge loss and Logistic loss. In Section 5, we propose efficient algorithms with linear time complexity for computing customer and product true size values that minimize the two loss variants. We show how our algorithms can be extended to leverage customer and product features, handle multiple personas within each customer account, and infer multi-dimensional true size vectors for customers and products in Sections 6-8. We compare our proposed product size recommendation algorithms to baselines on real-life Amazon Shoes data in Section 9, and offer concluding remarks in Section 10.

2 RELATED WORK

There is a vast body of research on latent factor models. The most predominant techniques are based on matrix factorization. [1, 2, 14]. Subsequently, there have been many extensions such as [5] to model general non-gaussian distributions, [6] based on tensor factorization, [4, 6] for dynamic extensions to capture varying user preferences, [7, 8] based on bilinear models, among many others. There have also been Bayesian approaches for recommendation systems [3, 9, 10], and techniques that leverage additional information like product meta data, user reviews and so on [12, 13].

The above-mentioned prior work on the general recommendation problem is based on learning a context independent representation. It learns latent factors with information aggregated across multiple contexts. The size recommendation problem is a special case of the general recommendation problem, where the only context of interest is the semantics of physical size of customers and products in a transaction, and its relationship to the physical fitment provided by the customer ($\{\texttt{Small}, \texttt{Fit}, \texttt{Large}\}$). Clearly, the general recommendation problem is not suitable for size recommendation as the learnt context independent representation has potential loss in information due to aggregation across contexts (such as personas).

We also note that there has been research on context aware recommender systems such as [17, 18], tensor factorization based recommender systems such as [15, 16] and physical context based recommender systems such as [19]. These techniques are not suitable for size recommendation that involves recommending a single precise size that best fits the customer, and leveraging ordinal information such as the catalog size ordering of products and customer returns data indicating product fitment for the customer. The physical context in these approaches refers to the current state of the user and the environment such as weather, humidity, location etc., but are not suitable to model the physical context of size and its relationship to the customer return data.

Our proposed model is also closely related to paired comparison models such as the Thurstone model [21] and Bradley-Terry model [20], which deal with modeling the outcomes of comparison between pairs of entities. These models are often used in psychometrics for comparing choices and in sports ratings (e.g., Elo ratings for chess) for comparing player performances. As in the current problem formulation, the goal is to estimate a latent factor (e.g., goodness of a choice, player's ability) associated with the entities such that the difference in the latent factor values determines the comparison outcome. However, the latent factors in traditional paired comparison models are only single dimensional unlike the proposed model that allows multi-dimensional latent vectors. The proposed model also extends the paired-comparison models by allowing weighting along the different dimensions, inclusion of feature effects, and handling sub-entities (e.g., personas within a customer account).

3 SIZE RECOMMENDATION PROBLEM

A product in categories such as shoes or apparel has multiple sizes. We will refer to the product as *parent* product and the different size variations as *child* products of the parent product. We will denote the set of all child products by $\mathcal{P}$ and the set of all customers by C. We have access to the set of past customer transactions $\mathcal{D}$ where each transaction in $\mathcal{D}$ is a triple of the form (customer, child product, return code). Specifically, each transaction has the form (i, j, y_{ij}) where i is the customer making the purchase, j is the purchased child product, and y_{ij} is either `Fit` if the customer does not return the product, or the return code provided by the customer when returning the purchased product, which can be `Small` or `Large`. For each child product j, we also have access to the catalog size c_j provided by the manufacturer.

In this paper, our objective is to recommend for a (customer, parent product) pair, the child product with the best size fit. We do this by inferring the true (latent) sizes of customers and child products using past transactions in $\mathcal{D}$. Let s_i denote the true size for customer i and t_j be the true size for child product j. Once we have the true sizes, we recommend the child product j whose true size t_j is closest to the customer's true size s_i. To simplify the presentation, in the remainder of the paper, we will refer to child products as simply products.

As mentioned earlier, we leverage past customer transactions in $\mathcal{D}$ to deduce true sizes for customers and products. Intuitively, if $\mathcal{D}$ contains a transaction $(i, j, \texttt{Fit})$, then the true sizes s_i and t_j must be close, that is, $|s_i - t_j|$ must be small. On the other hand, if transaction $(i, j, \texttt{Small}) \in \mathcal{D}$, then the customer true size s_i must be much larger than the product true size t_j, or alternately, $s_i - t_j$ must be large. Similarly, if transaction $(i, j, \texttt{Large}) \in \mathcal{D}$, then the customer true size s_i must be much smaller than the product true size t_j, or alternately, $t_j - s_i$ must be large. Let $f_w(s_i, t_j) = w \cdot (s_i - t_j)$ be our (linear) model with weight parameter $w \geq 0$. Furthermore, let $L(y_{ij}, f_w(s_i, t_j))$ be a loss function that takes on large values (and thus penalizes s_i, t_j values) for transactions (i, j, y_{ij}) such that: (1) $y_{ij} = \texttt{Fit}$ and $|f_w(s_i, t_j)|$ is large (exceeds a threshold value), (2)

y_{ij} = Small and $f_w(s_i, t_j)$ is a large negative quantity (falls below a threshold value), or (3) y_{ij} = Large and $f_w(s_i, t_j)$ is a large positive quantity (exceeds a threshold value). Then, what we seek are true size values s_i, t_j that minimize the loss function over transactions in $\mathcal{D}$.

The size recommendation problem can thus be formally stated as follows:

Problem Statement: Given past customer transactions $\mathcal{D}$, catalog sizes $\{c_j\}$ for products, and loss function $L(y_{ij}, f_w(s_i, t_j))$ for transactions $(i, j, y_{ij}) \in \mathcal{D}$, compute true sizes $\{s_i\}$ for customers and $\{t_j\}$ for child products such that $\mathcal{L} = \sum_{(i,j,y_{ij})\in\mathcal{D}} L(y_{ij}, f_w(s_i, t_j))$ is minimized. □

In the following sections, we first define different loss functions $L(y_{ij}, f_w(s_i, t_j))$, and then present algorithms for computing true sizes and parameter values that minimize the loss functions. We initially assume single-dimensional size values, and later in Section 8, we show how our schemes can be generalized to handle multi-dimensional size vectors.

4 LOSS FUNCTIONS

The values of the outcome y_{ij} in each transaction are one of Small, Fit or Large. There is an inherent ordering among these values and so it is preferable to model our loss function minimization problem as an ordinal regression problem. Li and Lin [11] present a detailed discussion on deriving loss functions appropriate for outcomes with multiple discrete ordered levels by reducing the ordinal regression problem to multiple binary classification problems. The construction expresses loss function L with ordinal outcomes Small, Fit and Large in terms of a new loss function L^{bin} with binary outcomes in $\{-1, +1\}$.

$$L(y_{ij}, f_w(s_i, t_j)) = \begin{cases} L^{bin}(+1, f_w(s_i, t_j) - b_2) & \text{if } y_{ij} = \texttt{Small} \\ (L^{bin}(-1, f_w(s_i, t_j) - b_2) & \\ +L^{bin}(+1, f_w(s_i, t_j) - b_1)) & \text{if } y_{ij} = \texttt{Fit} \\ L^{bin}(-1, f_w(s_i, t_j) - b_1) & \text{if } y_{ij} = \texttt{Large} \end{cases} \tag{1}$$

Above b_1 and b_2 are threshold parameters with $b_2 > b_1$ that split the model score $f_w(s_i, t_j)$ line into three segments such that a model score greater than b_2 corresponds to Small, a score less than b_1 corresponds to Large, and scores in between b_1 and b_2 correspond to Fit. As a result, if y_{ij} = Small then we want the model score $f_w(s_i, t_j)$ to be greater than b_2 and so we assign a positive class label (+1) to $f_w(s_i, t_j) - b_2$. On the other hand, if y_{ij} = Fit then we want the model score $f_w(s_i, t_j)$ to be less than b_2 and so we assign a negative class label (−1) to $f_w(s_i, t_j) - b_2$.

We can use any of the binary loss functions such as Hinge loss $L^{hinge}(y, \hat{y}) = \max\{0, 1 - y \cdot \hat{y}\}$ or Logistic loss $L^{logistic}(y, \hat{y}) = \log(\frac{1}{1+e^{-y\cdot\hat{y}}})$ for the binary loss function L^{bin}. Substituting L^{hinge} for L^{bin} in Equation (1) above, we get the following loss function values for Hinge loss:

ALGORITHM 1: Algorithm for computing customer and product true sizes.

ComputeTrueSizes(C, $\mathcal{P}$, $\mathcal{D}$, $\{c_j\}$)

1: **for each** product j, $t_j = c_j$.
2: $w = 1$; $b_1 = -1$; $b_2 = +1$.
3: **while** (not converged) and (numIterations < maxIterations) **do**
4: **for each** customer i, $s_i = \arg\min_{s_i}(\mathcal{L}|\{t_j\}, w, b_1, b_2)$.
5: **for each** product j, $t_j = \arg\min_{t_j}(\mathcal{L}|\{s_i\}, w, b_1, b_2)$.
6: $w, b_1, b_2 = \arg\min_{w,b_1,b_2}(\mathcal{L}|\{s_i\}, \{t_j\})$.
7: **end while**
8: **return** $\{\{s_i\}, \{t_j\}, w, b_1, b_2\}$.

$$L(y_{ij}, f_w(s_i, t_j)) = \begin{cases} \max\{0, 1 - f_w(s_i, t_j) + b_2)\} & \text{if } y_{ij} = \texttt{Small} \\ (\max\{0, 1 + f_w(s_i, t_j) - b_2)\} & \\ +\max\{0, 1 - f_w(s_i, t_j) + b_1)\}) & \text{if } y_{ij} = \texttt{Fit} \\ \max\{0, 1 + f_w(s_i, t_j) - b_1)\} & \text{if } y_{ij} = \texttt{Large} \end{cases} \tag{2}$$

The Hinge loss function values for Fit, Small and Large transactions with respect to customer true size s_i are shown in Figures 1, 2 and 3, respectively.

Similarly, substituting $L^{logistic}$ for L^{bin} in Equation (1) above gives us the following loss function values for Logistic loss:

$$L(y_{ij}, f_w(s_i, t_j)) = \begin{cases} \log(\frac{1}{1+e^{-f_w(s_i,t_j)+b_2}}) & \text{if } y_{ij} = \texttt{Small} \\ (\log(\frac{1}{1+e^{f_w(s_i,t_j)-b_2}}) & \\ +\log(\frac{1}{1+e^{-f_w(s_i,t_j)+b_1}})) & \text{if } y_{ij} = \texttt{Fit} \\ \log(\frac{1}{1+e^{f_w(s_i,t_j)-b_1}}) & \text{if } y_{ij} = \texttt{Large} \end{cases} \tag{3}$$

5 ALGORITHMS FOR COMPUTING TRUE SIZES

In this section, we present algorithms for computing true size values $\{s_i\}$ for customers and $\{t_j\}$ for products (as well as the optimal scaling factor w, and the thresholds b_1, b_2) that minimize $\mathcal{L} = \sum_{(i,j,y_{ij})\in\mathcal{D}} L(y_{ij}, f_w(s_i, t_j))$ for different loss functions such as Hinge Loss and Logistic Loss.

5.1 Hinge Loss

We consider Hinge loss because it can be optimized more efficiently and it also provides an upper bound on the zero one loss resulting in good generalization error bounds. Our algorithm (see Algorithm 1) computes true size values in an iterative manner with each iteration split into three phases that focus on computing customer true sizes while keeping product true sizes and parameter values fixed (Phase 1), computing product true sizes while keeping customer true sizes and parameter values fixed (Phase 2), and computing parameter values while keeping product and customer true sizes fixed (Phase 3). We describe the three phases in detail below.

For each product j, the true size t_j is initialized to the catalog size c_j (e.g. provided by the manufacturer). Furthermore, the thresholds b_1 and b_2 are initialized to -1 and +1, respectively, and the scaling factor w is initialized to 1. The algorithm terminates once the true

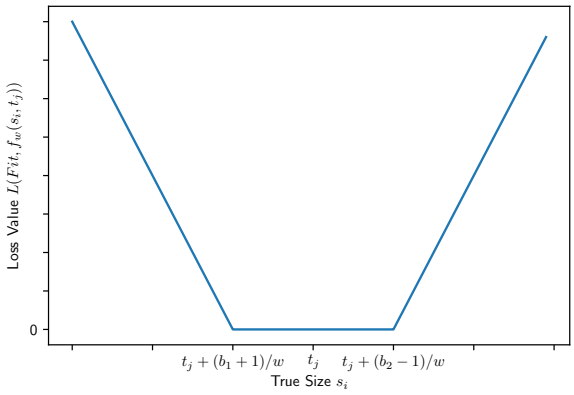

Figure 1: Hinge loss value for a `Fit` transaction vs s_i.

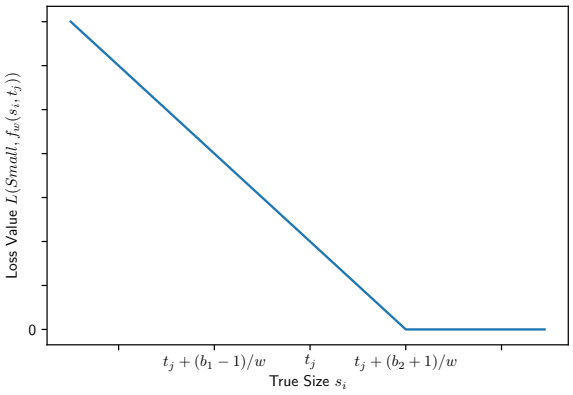

Figure 2: Hinge loss value for a `Small` transaction vs s_i.

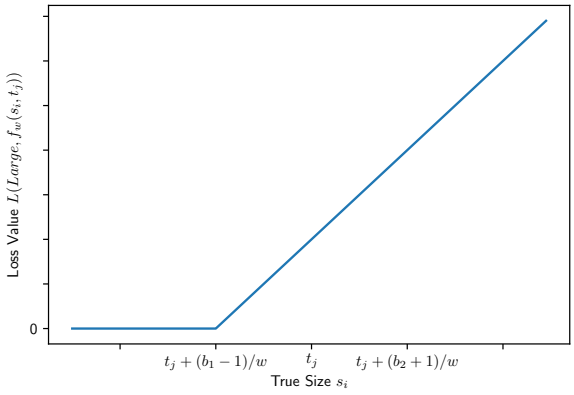

Figure 3: Hinge loss value for a `Large` transaction vs s_i.

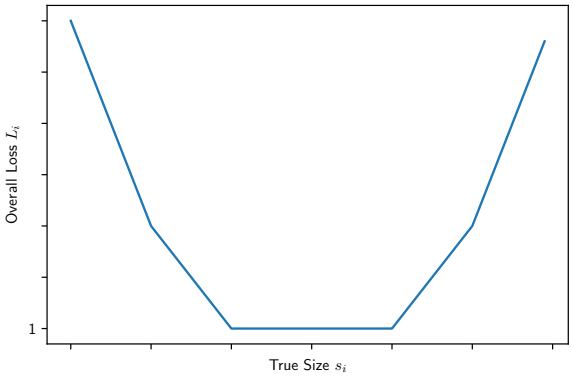

Figure 4: Illustrative overall hinge loss vs s_i.

size values converge or the number of iterations exceeds the max number of iterations.

Phase 1: Computing customer true sizes $\{s_i\}$. Figure 4 shows how the loss function $L(y_{ij}, f_w(s_i, t_j))$ value varies with s_i for Hinge loss (see Equation (2)). As can be seen, the slope of the loss function changes at $t_j + \frac{b_1-1}{w}$ for $y_{ij} = \texttt{Large}$ (Figure 3), at $t_j + \frac{b_1+1}{w}$ and $t_j + \frac{b_2-1}{w}$ for $y_{ij} = \texttt{Fit}$ (Figure 1), and at $t_j + \frac{b_2+1}{w}$ for $y_{ij} = \texttt{Small}$ (Figure 2). Furthermore, the loss function is convex and its slope only increases as s_i is increased.

Now, lets look at the overall loss function $\mathcal{L} = \sum_{(i,j,y_{ij})\in\mathcal{D}} L(y_{ij}, f_w(s_i, t_j))$. This does not have any (direct) coupling between customer sizes s_i and so s_i values that minimize the overall loss function can be computed independently. Specifically, for each s_i, we seek the size value that minimizes $\mathcal{L}_i = \sum_{(i,j,y_{ij})\in\mathcal{D}} L(y_{ij}, f_w(s_i, t_j))$ which is the sum of loss function values $L(y_{ij}, f_w(s_i, t_j))$ over products j purchased by i. Thus,

$$\begin{aligned}\mathcal{L}_i = &\sum_{(i,j,y_{ij})\in\mathcal{D}\wedge y_{ij}=\texttt{Small}} \max\{0, 1 - f_w(s_i, t_j) + b_2\} \\ &+ \sum_{(i,j,y_{ij})\in\mathcal{D}\wedge y_{ij}=\texttt{Fit}} (\max\{0, 1 + f_w(s_i, t_j) - b_2\} \\ &\qquad + \max\{0, 1 - f_w(s_i, t_j) + b_1\}) \\ &+ \sum_{(i,j,y_{ij})\in\mathcal{D}\wedge y_{ij}=\texttt{Large}} \max\{0, 1 + f_w(s_i, t_j) - b_1\}\end{aligned}$$

We now present an efficient scheme with time complexity linear in the size of $\mathcal{D}$ for computing the optimal s_i value that minimizes $\mathcal{L}_i$. Let

$$\begin{aligned}\mathcal{S}_i = &\{t_j + \frac{b_2 + 1}{w} | (i, j, y_{ij}) \in \mathcal{D} \wedge y_{ij} = \texttt{Small}\} \\ &\cup \{t_j + \frac{b_1 + 1}{w}, t_j + \frac{b_2 - 1}{w} | (i, j, y_{ij}) \in \mathcal{D} \wedge y_{ij} = \texttt{Fit}\} \\ &\cup \{t_j + \frac{b_1 - 1}{w} | (i, j, y_{ij}) \in \mathcal{D} \wedge y_{ij} = \texttt{Large}\}\end{aligned}$$

Observe that $\mathcal{L}_i$ is continuous, piecewise linear in s_i and convex (since it is the sum of convex loss functions). Figure 4 shows the value of $\mathcal{L}_i$ as the value of s_i is varied. As can be seen, the slope of $\mathcal{L}_i$ is non-decreasing as s_i is increased and transition points for the slope occur at s_i values in $\mathcal{S}_i$. Thus, the optimal s_i value that minimizes $\mathcal{L}_i$ can be determined by traversing the values in $\mathcal{S}_i$ in increasing order and selecting s_i to be the smallest value where the slope of $\mathcal{L}_i$ becomes non-negative.

Phase 2: Computing product true sizes $\{t_j\}$. Similar to customer true size estimation, for each product j, we need to find the size value that minimizes the sum of loss function values $\mathcal{L}'_j = \sum_{(i,j,y_{ij})\in\mathcal{D}} L(y_{ij}, f_w(s_i, t_j))$ over customers i who purchase j. Let

$$\begin{aligned}\mathcal{S}'_j =& \{s_i - \frac{b_2+1}{w} | (i,j,y_{ij}) \in \mathcal{D} \wedge y_{ij} = \texttt{Small}\} \\ &\cup \{s_i - \frac{b_1+1}{w}, s_i - \frac{b_2-1}{w} | (i,j,y_{ij}) \in \mathcal{D} \wedge y_{ij} = \texttt{Fit}\} \\ &\cup \{s_i - \frac{b_1-1}{w} | (i,j,y_{ij}) \in \mathcal{D} \wedge y_{ij} = \texttt{Large}\}\end{aligned}$$

The optimal t_j value that minimizes $\mathcal{L}'_j$ can be determined by traversing values in $\mathcal{S}'_j$ is increasing order and selecting the value where the slope of $\mathcal{L}'_j$ becomes non-negative.

Phase 3: Computing model parameter values $\{w, b_1, b_2\}$. Once the true sizes of customers and products are estimated, we fix them and estimate optimal values of the scaling factor w and thresholds b_1, b_2 that minimize $\mathcal{L}$ using any gradient descent algorithm (e.g. SGD).

5.2 Handling General Loss Functions

In the previous subsection, we presented an efficient algorithm for computing customer and product true size values when the loss function is Hinge loss. We can compute true size values $\{s_i\}$ for customers and $\{t_j\}$ for products that minimize $\mathcal{L}$ for other loss functions such as logistic loss (see Equation (3)) as well. Our algorithm is iterative, and in each iteration, alternates between computing s_i, t_j values and parameters w, b_1 and b_2. Specifically, in each iteration, the algorithm first uses gradient descent to compute s_i, t_j values with fixed w, b_1, b_2 values, and then uses gradient descent to compute w, b_1, b_2 values with fixed s_i, t_j values.

6 LEVERAGING CUSTOMER AND PRODUCT FEATURES

We can improve the accuracy of our model, combat data sparsity and handle cold start scenarios by leveraging additional information in the form of user features such as age, gender, Prime subscription, etc. and product features such as brand, category, product type, etc. Our new (linear) model with customer and product features x_{ij} is $f_w = w \cdot (s_i - t_j) + w_x \cdot x_{ij}$ – here w_x are the weights for features x_{ij}. Previous iterative algorithms for Hinge loss and Logistic loss can be extended to learn the weight parameters w_x for features as follows. In each iteration, the algorithms first compute s_i, t_j values (using the algorithms described in Sections 5.1 and 5.2 for Hinge loss and Logistic loss, respectively), and then use gradient descent to compute w, w_x, b_1, b_2 values with fixed s_i, t_j values.

ALGORITHM 2: Algorithm for computing customer persona and product true sizes.

ComputeTrueSizesPersonas(C, $\mathcal{P}$, $\mathcal{D}$, $\{c_j\}$)

1: **for each** product j, $t_j = c_j$.
2: $w = 1$; $b_1 = -1$; $b_2 = +1$.
3: **while** (not converged) and (numIterations < maxIterations) **do**
4: **for each** customer i **do**
5: Cluster purchase transactions (i, j, y_{ij}) of i in $\mathcal{D}$ into clusters $C_1, \ldots, C_k$.
6: **for each** cluster C_l, replace every transaction $(i, j, y_{ij}) \in C_l$ with $(i_l, j, y_{ij}) \in \mathcal{D}$.
7: **end for**
8: **for each** customer persona i_l, $s_{i_l} = \arg\min_{s_{i_l}} (\mathcal{L} | \{t_j\}, w, w_x, b_1, b_2)$.
9: **for each** product j, $t_j = \arg\min_{t_j} (\mathcal{L} | \{s_{i_l}\}, w, w_x, b_1, b_2)$.
10: $w, w_x, b_1, b_2 = \arg\min_{w, w_x, b_1, b_2} (\mathcal{L} | \{s_{i_l}\}, \{t_j\})$.
11: **end while**
12: **return** $\{\{s_{i_l}\}, \{t_j\}, w, b_1, b_2\}$.

7 HANDLING PERSONAS

In practice, a customer account may be shared by multiple individuals with different sizes, for e.g., different family members including children and adults. Thus, learning a single true size per customer account may lead to inaccurate true size estimates. In this section, we present a hierarchical clustering algorithm for identifying the personas within each customer account and grouping together the purchase transactions in $\mathcal{D}$ for each persona.

Algorithm 2 describes our algorithm for handling personas when computing customer and product true sizes. The algorithm is similar to Algorithm 1 except for the two points below:

- At the start of each iteration, for each customer i, the customer transactions $(i, j, y_{ij}) \in \mathcal{D}$ with `Fit` outcome are clustered based on the purchased product true sizes t_j. (The true sizes t_j are initialized to the catalog sizes c_j.) A hierarchical agglomerative clustering algorithm is used to cluster the customer's transactions. The algorithm iteratively merges clusters as long as the standard deviation of each merged cluster does not exceed a threshold σ_{max}. (σ_{max} is a hyper-parameter which is tuned by cross validation.) Let $C_1, \ldots, C_k$ be k clusters obtained as a result of clustering the `Fit` transactions belonging to customer i. Further, let m_l be the mean of t_j values for transactions in cluster C_l. Next, the `Small` and `Large` transactions (i, j, y_{ij}) of customer i are assigned to one of the k clusters as follows: (1) If y_{ij} = `Small`, then the transaction is added to cluster C_l where l is the index of the cluster with the smallest m_l value greater than t_j, and (2) If y_{ij} = `Large`, then the transaction is added to cluster C_l where l is the index of the cluster with the largest m_l value less than t_j. The k clusters $C_1, \ldots, C_k$ correspond to transactions of personas for customer i.
- After the clustering step, let $C_1, \ldots, C_k$ be k clusters corresponding to personas for customer i. We treat each persona with transactions in the l^{th} cluster C_l as a separate customer i_l. Thus, all transactions $(i, j, y_{ij}) \in \mathcal{D}$ belonging to

cluster C_l are replaced with transactions (i_l, j, y_{ij}) where i_l is the new customer corresponding to the persona for cluster C_l. The algorithms described in Sections 5 and 6 are then run on the new set of transactions belonging to customer personas to compute the true size values $\{s_{i_l}\}, \{t_j\}$ and parameters w, w_x, b_1, b_2.

8 EXTENSION TO MULTI-DIMENSIONAL SIZES

We can extend our techniques to infer multi-dimensional true size vectors for customers and products. The various dimensions capture different aspects related to size such as length and width for shoes, or waist and length for jeans. We define $f_{\mathbf{w}}(\mathbf{s_i}, \mathbf{t_j})$ for a 2 dimensional formulation as follows: $f_{\mathbf{w}}(\mathbf{s_i}, \mathbf{t_j}) = w_1(s_{i_1} - t_{j_1}) + w_2(s_{i_2} - t_{j_2})$, where w_1 and w_2 represent length and width scaling parameters, respectively, s_{i_1} and s_{i_2} represent customer length and width, respectively, and t_{j_1} and t_{j_2} represent product length and width, respectively.

We assume a transaction with physical fitment "Fit" corresponds to a product fitting the customer on both length and width dimensions. Correspondingly, the Algorithm 1 can be first applied to learn the customer true width s_{i_2} using only "Fit" transactions (ignoring step 6). Then a modified version of Algorithm 1 that (1) initializes the learnt customer width (s_{i_2}) and manufacturer provided catalog sizes (t_{j_1}, t_{j_2}), and (2) updates sizes s_{i_1}, s_{i_2}, t_{j_1} and t_{j_2} followed by model parameters w_1 and w_2 in that order using a procedure similar to Section 5.1. The above procedure can similarly be extended to any given d-dimensional customer and product vectors.

9 EXPERIMENTS

In order to measure the performance of our proposed models, we evaluated our techniques on offline Amazon Shoes datasets, and also carried out an online A/B test for the Mobile & Tablet experience. Our offline experimental setup is described in Section 9.1, while Section 9.2 provides details about the data and offline experimental results. The results from the online A/B test is noted in Section 9.3.

9.1 Offline Experimental Setup

For measuring performance of our models on offline datasets, we compare our results against a baseline model that uses the catalog size as a proxy for the latent size of a product and estimates a customers latent size as the average of all catalog sizes for that customer's transactions with a *fitSuitabilityCode* of `Fit`. However, note that the offline datasets are restrictive in nature for our setting. In such a setup, since our recommended sizes were never shown to the customer, we never get to know the customer's response to our recommendations. The only datapoint that we observe is the transaction that the customer carried out and the resultant *fitSuitabilityCode* of the transaction. Hence, in the offline setting we compare size recommendation schemes by their ability to predict the *fitSuitabilityCode* outcome of unseen transactions. We equate such an ability to the power of a classifier which takes the learnt latent product and customer sizes of the size recommendation algorithm as input and predicts the *fitSuitabilityCode* class of the transaction as the output. We carry out a time-based split of available customer transactions into training and test set to closely reflect online scenario where transactions up-to a certain time can be used for predicting future transactions. The size recommendation algorithm (ours and the baseline) learn customer and product true sizes on the training set. We also learn logistic regression (Linear) and Random Forest (RF) classifiers which then take the learnt customer and product latent sizes as input and produce *fitSuitabilityCode* as output. Performance of such Linear and RF classifiers on the test set are then used to compare the performance of our algorithm with that of the baseline.

9.2 Offline experiments with Softlines shoes data

Data Analysis: We collected transactions data for customers, transactions were represented in terms of the following attributes - *customerId*, *productId* and *fitSuitabilityCode* $\in$ {`Small`, `Fit`, `Large`}. We used the first 80% of transactions in temporal order for training while the remaining were used as the test set. We collected transactions data for customers for six product categories labeled A-F for proprietary reasons. The number of transactions in each dataset is shown in Table 2. Each transaction is represented in terms of the following attributes – customerID, productID, fitSuitabilityCode $\in$ {`Small`, `Fit`, `Large`}.

In our datasets, catalog size for all products varies between 4 and 20 on the US sizing scheme, with 0.5 size intervals. We plot the histogram of count of products and transactions respectively for different catalog sizes in Figures 5 and 6. In order to safeguard proprietary information, we do not report the actual product and transaction counts for different catalog sizes in Figures 5 and 6, respectively. While these plots tend to indicate that the customers may have a slight bias towards buying full catalog sizes over half sizes, but such behavior could be a manifestation of half sizes being out-of-stock at the time of purchase as well. And since we do not record in-stock availability of various catalog sizes at the time of a customer's purchase, we cannot pinpoint the exact root cause for the full sizes being more popular than half sizes, at least not based on the data that we have. Also, for categories except "C", most customer purchases are for catalog sizes between 7 and 10. For the "C" category, the popular catalog sizes are between sizes 8 and 12. The number of transactions is extremely small for sizes greater than 14 and is almost negligible for products with sizes more than 17.

We plot the histograms of number of products and customers respectively with different transactions counts in Figures 7 and 8. In order to safeguard proprietary information, we do not report the actual product and transaction counts for different catalog sizes in Figures 7 and 8, respectively. Note that a majority of products and customers have fewer than 3 transactions in the training data.

Offline Experimental Results: Table 1 summarizes the results of our experiments on various offline datasets. Performance of Algorithm 1 (described in Section 5.1) with and without personas are compared with the baseline method discussed in Section 9.1 (Baseline) and a baseline method which has information about personas (Baseline Persona). Logistic Regression (Linear) and Random Forest (RF) based classifiers are then built based on the output of each of the algorithms, and then their performance is compared on the basis of the weighted AUC of the resulting classifier, as discussed in Section 9.1. From Table 1, we can make the following inferences:

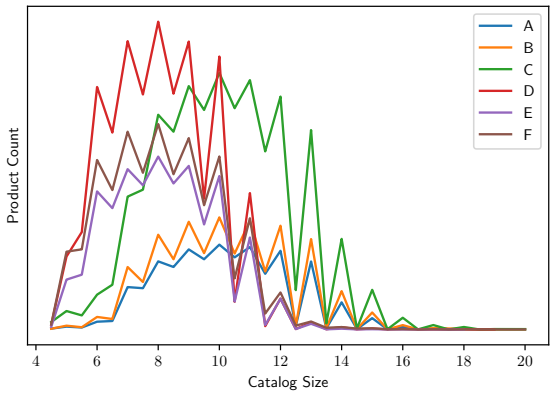

Figure 5: product count across catalog size.

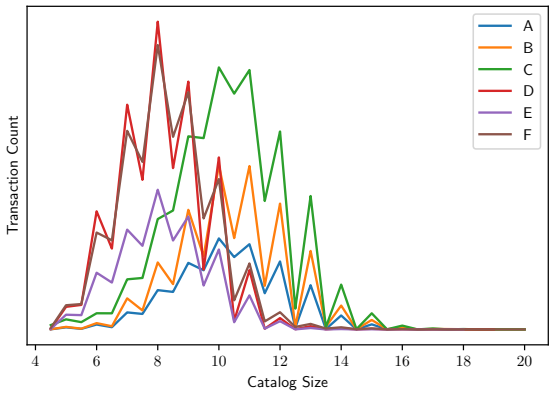

Figure 6: Transaction count across catalog size.

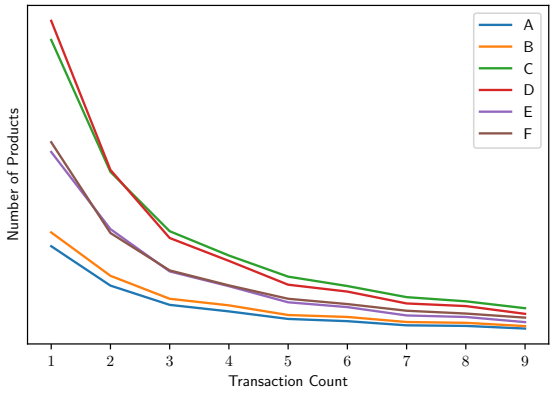

Figure 7: Histogram of products with transaction count.

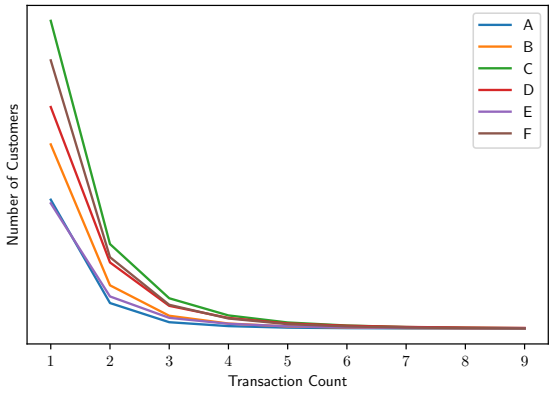

Figure 8: Histogram of customers with transaction count.

Dataset	Baseline RF	Baseline Persona Linear	Baseline Persona RF	Algorithm 1 Linear	Algorithm 1 RF	Algorithm 1 Persona Linear	Algorithm 1 Persona RF
A	0.6%	0%	0%	16.4%	17.2%	17.9%	**18.1%**
B	2.1%	0.4%	2.1%	20.3%	21.3%	**21.3%**	20.5%
C	3.8%	1.9%	1.9%	15.7%	16.1%	**18.4%**	17.8%
D	2.7%	1.2%	1.6%	20.0%	20.2%	**21.3%**	20.7%
E	1.5%	0.2%	0.6%	15.8%	15.6%	**17.4%**	**17.4%**
F	2.5%	2.7%	2.3%	18.1%	17.3%	**18.5%**	17.3%

Table 1: Summary of offline experimental results. We report the percentage improvements in weighted AUC over an above Baseline Linear AUC values for each dataset.

A	B	C	D	E	F
10.4M	17M	33.2M	25M	12.9M	27M

Table 2: Summary of transaction count for all datasets.

(1) Algorithm 1 has 17-21% higher AUC compared to the baseline method and (2) Modeling Personas boosts the AUC of Algorithm 1 further up-to 2.3%.

9.3 Online A/B test Results

Based on the output of Algorithm 1, we ran an online A/B test against an incumbent size recommendation algorithm as control in the US marketplace for Mobile and Tablet experience. The incumbent algorithm is a proprietary algorithm that groups products into clusters based on co-purchase frequency. Thus, product pairs that are bought by a large set of customers end up in the same cluster. A measure of central tendency based on the catalog sizes of products participating in each of the clusters is used to assign the cluster and its participating products a latent size. Recommendations to customers are then carried out in the latent size space. Conceptually, Algorithm 1 also tries to achieve a similar result, though in a more principled manner. The results of the online A/B test showed that we are able to achieve an improvement of 0.49% (p-value $<$ 0.001) in the percentage of `Fit` transactions over control, which also translates into a reduction of the same magnitude in the number of return transactions.

10 CONCLUSION

In this paper, we proposed a novel latent factor model for predicting product size fits based on the difference between customer and product (latent) true sizes. We devised efficient algorithms for computing customer and product true size values by minimizing two loss function variants, and proposed a hierarchical clustering algorithm to discover the underlying personas for each customer. On Amazon shoe datasets, our latent factor model has an AUC that is 17-21% higher compared to baselines. In an online A/B test, our algorithms show an improvement of 0.49% in percentage of Fit transactions over control.

REFERENCES

[1] Ruslan Salakhutdinov and Andriy Mnih. Probablistic Matrix Factorization. NIPS 2008.

[2] Yehuda Koren, Robert Bell and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. IEEE Computer 2009.

[3] P. K. Gopalan, L. Charlin, and D. Blei. Content-based recommendations with Poisson factorization. "In Advances in Neural Information Processing Systems", pages 3176-3184, 2014.

[4] Dynamic Poisson Factorization, by Laurent Charlin, Rajesh Ranganath, James McInerney and David M. Blei, ACM Conference on Recommender Systems, Recsys 15

[5] A Framework for Matrix Factorization based on General Distributions, by Josef Bauer and Alexandros Nanopoulos, ACM Conference on Recommender Systems, Recsys 14

[6] Modeling the Dynamics of User Preferences in Coupled Tensor Factorization by Dimitrios Rafailidis and Alexandros Nanopoulos, ACM Conference on Recommender Systems, Recsys 14

[7] Wei Chu and Seung-Taek Park. Personalized Recommendation on Dynamic Content Using Predictive Bilinear Models. WWW 2009.

[8] Deepak Agarwal and Bee-Chung Chen. Regression based Latent Factor Models. KDD 2009.

[9] Bayesian latent variable models for collaborative item rating prediction, Morgan Harvey, ACM International Conference on Information and Knowledge Management, CIKM 2011.

[10] Bayesian probabilistic matrix factorization using Markov chain Monte Carlo, Ruslan Salakhutdinov, Andriy Mnih, International Conference on Machine Learning, ICML 2008.

[11] Ling Li and Hsuan-Tien Lin. Ordinal Regression by Extended Binary Classification.Neural Information Processing Systems, NIPS 2006.

[12] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. ACM Conference on Recommender Systems, Recsys 2013.

[13] Meta-Prod2Vec – Product Embeddings Using Side-Information for Recommendation by Flavian Vasile, Elena Smirnova, Alexis Conneau, ACM Conference on Recommender Systems, Recsys 16

[14] Asynchronous Distributed Matrix Factorization with Similar User and Item Based Regularization, Bikash Joshi, Franck Iutzeler, Massih-Reza Amini, ACM Conference on Recommender Systems, Recsys 16

[15] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In Proceedings of the fourth ACM conference on Recommender systems (RecSys '10). ACM, New York, NY, USA, 79-86. DOI=http://dx.doi.org/10.1145/1864708.1864727

[16] Atsuhiro Narita, Kohei Hayashi, Ryota Tomioka, and Hisashi Kashima. 2011. Tensor factorization using auxiliary information. In Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part II (ECML PKDD'11), Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis (Eds.), Vol. Part II. Springer-Verlag, Berlin, Heidelberg, 501-516.

[17] Panniello, Umberto, et al. "Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems." Proceedings of the third ACM conference on Recommender systems. ACM, 2009.

[18] Adomavicius, Gediminas, and Alexander Tuzhilin. "Context-aware recommender systems." Recommender systems handbook. Springer US, 2015. 191-226.

[19] Karen Church, Barry Smyth, Paul Cotter, and Keith Bradley. 2007. Mobile information access: A study of emerging search behavior on the mobile Internet. ACM Trans. Web 1, 1, Article 4 (May 2007). DOI=http://dx.doi.org/10.1145/1232722.1232726

[20] Bradley, R.A. and Terry, M.E. (1952). Rank analysis of incomplete block designs, I. the method of paired comparisons. Biometrika, 39, 324–345.

[21] Thurstone, L.L. (1959). The Measurement of Values. Chicago: The University of Chicago Press.