FILTERING UNFIT CLOTHES FOR CUSTOMER

Sarah Ekaireb,Chang Zhang

University of California San Diego, La Jolla, CA 92093, USA

Abstract

In this paper we build a clothing fit predictor. We created a model that takes in a user and item, and predicts whether the item will fit, be too small, or be too large. Being able to predict whether a piece of clothing will fit is important, especially when a lot of people shop for clothing online (where there are no fitting rooms). Many clothing websites have some sort of fit predictor based on the user's body size information, size in different brands, etc. We experimented with four clothing fit models, a KNN model, a logistic regression model, a latent factor model, and a combined model (combination of logistic regression and latent factor). The combined model performed the best with a categorization accuracy of 0.7516 and an average AUC of 0.6870.

1. Dataset

We are using the *RentTheRunway* dataset [1]. The dataset consists of fit feedback from RentTheRunway, a site that rents out women's designer clothing for various occasions. For a given user-item pair, the user has rated the item as either small, fit, or large. In addition, each review contains the review text, user's size information, rating, rental occasion, etc.

In Table 1 we provide some general statistics about the dataset. One notable observation is that the majority of users have only one review. This means that there will be a lot of cold-start cases to handle during testing.

Statistic	
# reviews	192544
# users	105571
# items	5850
average # ratings per user	1.82
average # ratings per item	32.91
# users with 1 review	71824
# items with 1 review	341

Table 1: General statistics

Clothing fit can be very subjective. What one user may view as too large, another user may view as an oversized style. This poses a challenge when determining what review information to use to build out models.

There are multiple fields relating to the user's body size including bust size, body type, weight, and height. All of this size information can change and fluctuate over time; for instance someone may gain or lose weight. However, it is likely that there will not be any drastic changes within a short period of time. In Table 2 we show the fraction of reviews that contain each of these features.

Feature	**Fraction**
weight	0.84
height	0.99
bust size	0.90
body type	0.92
all above features	0.76

Table 2: Fraction of reviews with each feature

In addition, one of the fields is size (clothing size). Many users rented fitting items of different sizes from the same category (for instance a size 4 and size 8 dress). In a review, a user even writes "...I'm anywhere between a 0 and a 4 depending on the brand...", while they put down "14" for size. Due to the differences in sizes across different items and brands, it may be challenging to use the size as a feature in certain types of models. However, since one item can have multiple sizes, this is an important feature in determining fit.

Statistic	**Number**	**Fraction**
small	25779	0.13
fit	142058	0.73
large	24707	0.12

Table 3: Fit statistics

In Table 3 we provide some fit statistics about the dataset. There is a slight class imbalance, with the majority of reviews being of the class "fit". We experimented with methods to overcome this imbalance including training with an equal amount of data from each class. In the future, we could also experiment with creating more small and large data points using the fit information given. For instance, if we know for a particular item a user fits into a size 4, then a size 8 will be too large for them. To do this we would need to include the clothing size as a feature or input in our model.

2. Predictive Task

Given a user-item pair, we would like to predict how the item will fit the user. There are three fit classes: small, fit, and large. The fraction of reviews belonging to each class can be seen in Table 3 above.

We evaluate our models and access the validity of our models' predictions by using categorization accuracy and area under the ROC curve (AUC). The categorization accuracy measures what fraction of the time the correct class is predicted.

$$\text{Categorization Accuracy}(\hat{r}, r) = \sum_{u,i} \delta(\hat{r}_{u,i} = r_{u,i})$$

We can not solely use categorization accuracy to evaluate our model because it does not take into account the rates of false positives and false negatives. Also, it does not take into account the fact that our dataset is imbalenced, so a high accuracy may not be representative of how the model is actually performing.

Hence, we also need to use the AUC as a form of evaluation. The AUC is a metric that measures how well a model performs on different sets of classes. It is a particularly useful metric when the dataset is imbalanced. A perfect model would have an AUC of 1 and a model that outputs random results would have an AUC of 0.5.

In section 1, we wrote about some of the features that we found useful. For our logistic regression models, we primarily use the fit feature. We also experimented with the user body size features and size features. We found that the other features in the dataset (review text, review summary, rating, etc.) to not be very useful in the models we built. For the latent factor model, we do not use any feature engineering, as a low-dimensional representation of the users and items is found through minimizing the objective function. However, we do use height and weight as features when handling cold cases when making predictions using the latent factor model.

Baseline

We built a K-nearest neighbors (KNN) model as our baseline. For each user-item pair, we find the $K = 6$ nearest users that bought that particular item (from the training set), and choose the max fit among those users. In Figure 1 we show a visualization of this process.

For each user, the feature vector consists of the user's height, weight, and size. We measure the distance between two users by using a weighted euclidean distance. We weight each of the features slightly by how much we believe a change in that feature would reflect a change in fit. We believe this a good baseline because it takes into account the most similar users that have bought the same item in the past, and how that item fit them. There are some issues with scalability of this model because as the number of users that bought a particular item increases, the search time to find the KNN also increases.

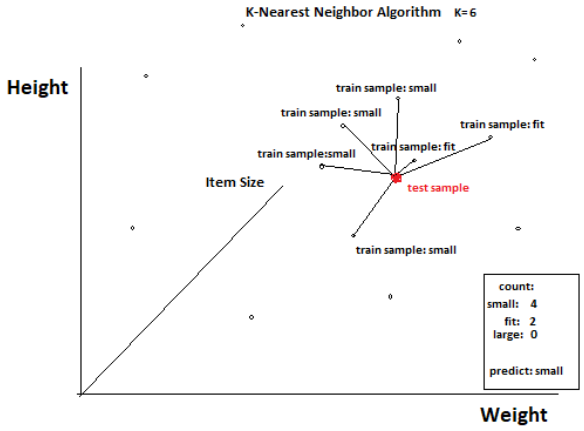

Fig. 1: Baseline 6-NN visualization

3. Related Work

The customer size recommendation problem is a relatively recent subject with only a handful of published studies. One of the early successful studies modeled the size recommendation problem as a binary classification problem. [2] The classifiers leveraged the product measurements and observed user information in addition to latent user and item vectors to achieve a high-precision model. Since each classifier has to be trained on an item category, a large portion of the samples was filtered, which leads to a large low number of usable data for training as each sample has to have at least two previous purchases in that category. Another issue with this model is that it has a low coverage percentage. A more generalizable model is needed for cases where we would like to recommend users items from categories they have not previously made purchases.

A more recent paper solves the size recommendation problem by approximating a fit function, a function that returns how well an item fits a user, using past purchase history. [3] This function relies on two important latent variables: true user size and true item size. These two variables are learnable by optimizing a loss function. The loss function is based on the assumption that if an item is labeled as large by a user, the actual size of the item is a lot larger than the actual size of the item and vice versa. They used two thresholds to ensure that the loss function will penalize only if the difference between the user fit-size and the item fit-size is large. Since the thresholds are non-learnable parameters, the model can be hard to tune as we have to search for the thresholds that would work well. Fine-tuning these thresholds can then lead to potential overfitting. If user rating behavior changes

slightly, thresholds must be readjusted.

Another recent paper, similar to the above study, trained and tested on the *RentTheRunWay* dataset (the dataset they contributed and we use in our study), finds latent variables by utilizing the customers' fit feedback which captures the customer preferences on product attributes such as shoulders or waist sizes. They recognized one of the challenges for the size recommendation problem is the imbalance of the labels. [1] Instead of classifying using linear regression, their best-performing proposed model used k-nearest-neighbors with a learned metric that aims to minimize the distances of transactions of the same class and maintain the margin of transactions of different classes. Because this model uses the loss function from the above study, it is also harder to tune. Comparing the performance of our model on the *RentTheRunWay* dataset with theirs, we see that our model performance is worse than their best-performing model but it is a slight improvement over their second-best model. Similar to our findings, they also saw a decrease in AUC score when predicting samples that are labeled *fit*.

A state-of-the-art neural-network architecture can also be developed for the size recommendation task. [4] SF_{NET} was designed to harness the potential of both the collaborative methodology and the content-based approach. Item and user properties in the *RentTheRunWay* dataset were concatenated with the embedding features to feed into the SF_{NET} and a softmax function was attached to the last layer to generate fit and size prediction. This proposed approach alleviates the issue of data sparsity and the results have shown improvement over the previous study when tested on the *RentTheRunWay*. Comparing our model with the state-of-the-art model, we see that the stat-of-the-art model is a significant improvement over our model. Yet, it suffers the same issue from class imbalance.

ModCloth is another popular dataset for measuring the performance of a size recommendation model. It was collected by [1] and used by [1] and [4]. The results from *ModCloth* are comparable with the *RentTheRunWay*. We only used *RentTheRunWay* which is enough for analyzing the performance of our model.

4. Models

We first built a latent factor model. Based on the model's performance and what we learned in the process, we then built a logistic regression model with the features we thought would be the most useful. To make a prediction, we use both the logistic regression model and the latent factor model to predict the probability of a sample being *small*, *fit*, or *large*. We then take the weighted average of the normalized probabilities from the two models to make a combined prediction. This will ideally combine the strengths of both models, which will be described below. The weights are set based on the AUC score of each model. The combined model is described in Figure 2.

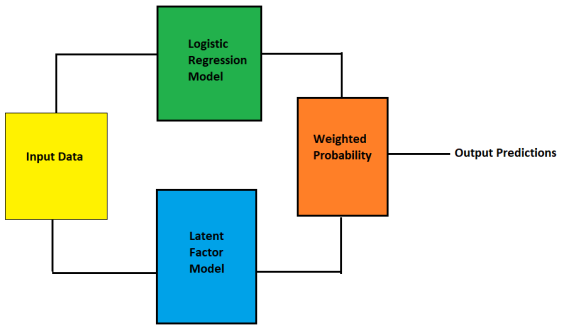

Fig. 2: Model description

In the following sections we will describe the logistic regression and latent factor models that are used to create the combined model.

Logistic Regression Model

We built a logistic regression model using features that only include item fit information from the training dataset. This model does not use any of the user-specific, user-item interaction, or temporal information. The features for a particular item are the percentage of past reviews that are "small", the percentage of past reviews that "fit", and the percentage of past reviews that are "large".

We chose to use these as our features because clothing items can run small, be true to size, or run large. When a piece of clothing runs small it means that the item of clothing runs smaller than clothing of the same size from most other brands. Similar statements apply for when a piece of clothing is true to size or runs large. Our hope is that the model is able to determine whether a piece of clothing typically runs small, is true to size, or runs large by using the past class percentages. We attempted to add in other user related features to this model, but did not see an improvement in performance in doing so.

One drawback to this feature representation, is that we are assuming the user is ordering their typical size. If users are not ordering their typical sizes this could lead to incorrect predictions.

There are three classes (fit, small, and large), so we used the many versus rest approach for multi-class classification. Essentially, we broke the problem into three binary classification problems: fit vs not fit, small vs not small, and large vs not large. We have a logistic regression model for each of the binary classification problems. To make a prediction on a user-item pair, we run all three of the classification models and choose the class (fit, small, or large) with the highest probability.

The training for this model was relatively fast. And, there are no issues with scalability as the number of users and items increase. For this model we did not experience any overfitting.

Latent Factor Model

The fit feedback is an ordinal feature of the dataset. Clothing that is too small has to be smaller than clothing that fits just right, which is smaller than the one that is too large. We can convert it to a real-valued feature and this allows us to approximate a real-valued "fitness" function by using gradient-based optimization algorithms. We map "small" to value 1, "fit" to 3, and "large" to 5. We can predict the fit of a given user-item pair by passing into the "fitness" function and the fit is whichever one the closest to the output (either 1, 3, or 5) is mapped to. The "fitness" function is described in the following equation:

$$f_{fit}(user, item) = \alpha + \beta_{user} + \beta_{item} + \gamma_{user} \cdot \gamma_{item}$$

where α is the global bias term (we set it as the average fitting score), β_{user} and β_{item} are the bias terms that hold the concept some items tend to have a looser fit than the average and some users tend to have less restrictive fit preferences than the average, and vice versa. The γs are the k-dimensional latent features (we chose k to be 5).

Finding the optimal β and γ is to solve the following optimization problem:

$$argmin_{\beta\gamma} = \sum_{user,item}(f_{fit}(user, item) - R_{user,item})^2 + \lambda_1[\sum_{user}\beta_{user}^2 + \sum_{item}\beta_{item}^2] + \lambda_2[\sum_{user}||\gamma_{user}||_2^2 + \sum_{item}||\gamma_{item}||_2^2]$$

where λ_1 and λ_2 controls the strength of regularization on parameter β and γ respectively. This optimization problem is solved using the Adam optimizer. In each iteration of optimization, 60000 samples were sampled evenly from the three classes and they were used to train the parameters. The process of optimization stops at 300 iterations as we would like to avoid model overfitting. However, since our samples were drawn randomly, the test results from each training run are inconsistent to some extent, and it is hard to pinpoint the best hyperparameters for our latent factor model. Nevertheless, we searched for the best hyperparameter using a coarse grid search for maximizing the effectiveness of the hyperparameter search.

While training the latent factor model, we noticed that, for each item, there are multiple sizes that could be purchased by users. In the earlier model, we failed to take into consideration of the item sizes. To fix this issue, we created unique β and γ for each item and size pair.

Handling cold-start cases

We experimented with some new methods of handling cold-start cases. Around 68 % of users only provided one review in the dataset. This means over half of the test samples are cold-start cases where we have no user history in the training set. to solve the cases where a test user does not exist in the training set, we use the nearest-neighbor approach (shown in figure 3). We take the weight and height of the unseen user and find the most similar user in the training set and use the parameters of that user to predict the level of fitness. If the unseen user has not entered their weight and height, we therefore cannot find the most similar user. For this case, we predict *fit* (because this is the most common class). One edge case is that the item in the test dataset does not exist in the training set. Since we have no knowledge of this new item, we also predict *fit*.

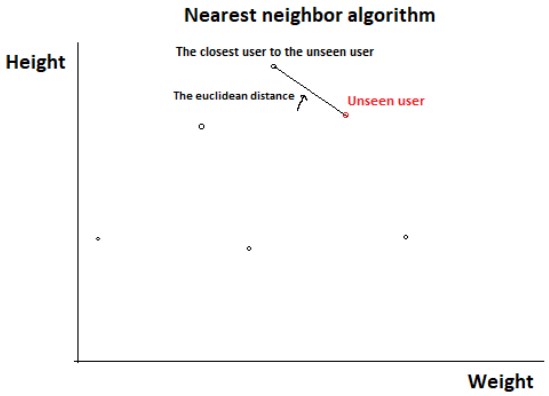

Fig. 3: User selection using nearest neighbor

5. Results

Our results are summarized in Table 4. We found the combined model (described in the Model section) performed the best out of all the models. We believe that this model performed the best because it has the strengths of both the the Logistic Regression Model and Latent Factor Model. The Logistic Regression Model captures whether an item of clothing runs small, true to size, or large. And, the Latent Factor model captured a feature representation pertaining to the users and sized items.

Model	Cat. Accuracy	Avg. AUC
Baseline (KNN)	0.7268	0.6423
Logistic Regression	0.7500	0.6771
Latent Factor	0.7159	0.6328
Combined	0.7516	0.6870

Table 4: Model Results

In Figures 4-6 we show the ROC curves for each of our models. These plots show the ROC curve of each class, and the AUC for each class.

One observation is that the area under the curve is the least for the class "fit" in all three of the models. This is consistent with the results in [1] [4]. This shows that in general, the models are better at distinguishing between small vs. not small, and large vs. not large, than the models are at distinguishing between fit vs. not fit. We believe that this may be due to the class imbalance and large overlapping decision boundaries.

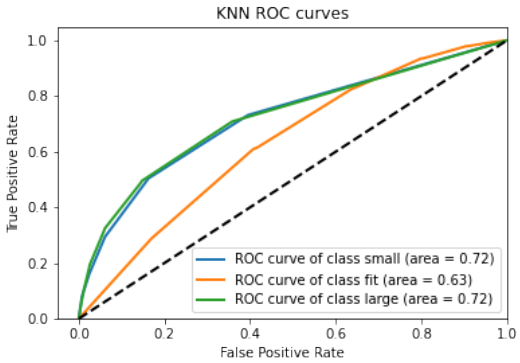

Fig. 4: Baseline 6-NN AUC Plots

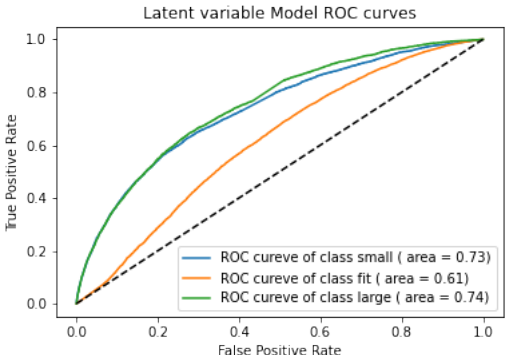

Fig. 5: Latent factor AUC Plots

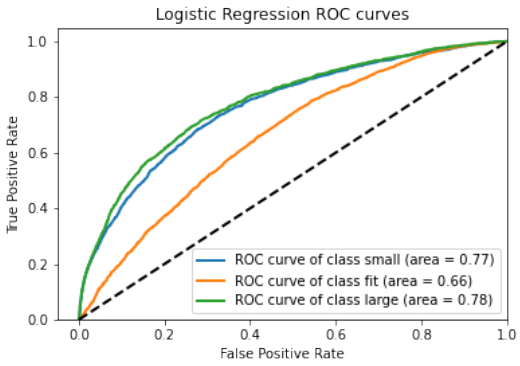

Fig. 6: Logistic Regression AUC Plots

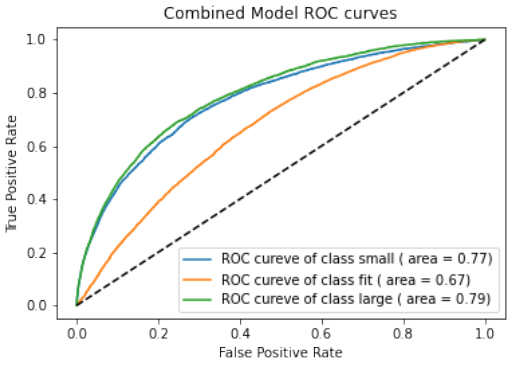

Fig. 7: Combined Model AUC Plots

The Logistic Regression model outperformed the baseline 6-NN model and the Latent Factor model. The logistic regression model solely uses the "fit" feature of previous reviews for the features. As mentioned before, we chose to use these features because how the item fits (small, true to size, large) is likely the same for the majority of users. This model performed well, showing that our theory about the sizing may be true, and that this may be a larger factor in determining fit that some of the other features.

The latent factor model performed worst than the logistic regression model because of two reasons. First, the latent factor model requires more extensive tuning as it has more hyperparameters and it is prone to overfit during training. Second, the latent factor model assumes that a user will rate the item *small* or *large* if they are smaller or larger than the user's true size. However, in the case of the dataset, users rate the item sizes base on whether the item's size is similar to the user's expectation. If the item is larger than the user's true size, the user still might rate the item as fit ("true to size") because the user expects the item to be larger than their body. Given more time, we believe we would be able to improve our latent factor model slightly with more tuning. Improving the latent factor model would likely also improve the combined model.

1. REFERENCES

[1] Rishabh Misra, Mengting Wan, and Julian McAuley. Decomposing fit semantics for product size recommendation in metric spaces. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 422–426, New York, NY, USA, 2018. Association for Computing Machinery.

[2] G. Mohammed Abdulla and Myntra Designs. Size recommendation system for fashion e-commerce. 2017.

[3] Vivek Sembium, Rajeev Rastogi, Atul Saroop, and Srujana Merugu. Recommending product sizes to customers. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 243–250, New

York, NY, USA, 2017. Association for Computing Machinery.

[4] Abdul-Saboor Sheikh, Romain Guigourès, Evgenii Koriagin, Yuen King Ho, Reza Shirvany, Roland Vollgraf, and Urs Bergmann. A deep learning system for predicting size and fit in fashion e-commerce. *CoRR*, abs/1907.09844, 2019.