"Rent The Runway" Latent-Factor Model

Names: Asher Av, Brady Zhou, Sean Ng, SungJun Kim

Introduction

With the growth of online shopping over the past few years, personalized fashion recommendation is becoming more and more important. Especially since online shopping does not offer fitting rooms, having good recommendations are even more crucial. Companies such as Rent The Runway want to give accurate and personalized recommendations to their users. Using the data from Rent The Runway, we explored the data looking at distributions and summary statistics and then built a latent factor model to recommend personalized clothing items to users.

Exploratory Data Analysis

In our exploratory data analysis we got an overview of some of the basic statistics of the dataset. Some of the features of the dataset include: fit, user id, bust size, item id, weight, rating, occasion that the garment is rented for, reviews, the renter's body type, category of the garment, height of the user, size of the garment, age of the user, and the review date. The following table is a statistical summary of some of the numeric features of the dataset, which includes 192544 observations (called interactions) and 15 features.

We looked at the distribution of height. We found that the height resembled a Gaussian distribution. At the extremes, the minimum height was 54-inches or 4'6" and the maximum height was 78-inches or 6'6". And, with a reported 191,867 users, the average height was about 65 inches or 5'5" and the most reported height was 64 inches or 5'4".

We found that the weight feature only somewhat followed a Gaussian distribution. At the extremes, the minimum weight was 50-lbs and the maximum weight was 300-lbs. With a reported 162,562 users, the average weight was 137-lbs and the most reported weights were between the interval of 133-lbs to 141- lbs. The weight feature thus yielded a positive skew and so users across the dataset tended to report a weight closer to the lower extreme.

We explored the percentage of missing values from each of our 15 features. 3 features in particular had a percentage greater than 1. The feature with the highest missing percentage was weight at 15%. The feature with the second highest missing percentage was bust size at 9%. The third highest missing percentage was body type at 7%.

We also explored the average ratings across the years on the dataset. Since there was little data outside the years of 2011 to 2018, we sampled the average ratings at 2-month intervals within those years and found that users tended to report higher ratings as the years went by.

We explored the interactions, and in particular the single-interactions within the dataset. We found that the percentage of single-interactions within the dataset was relatively high at 68%. This made us realize that our model may have issues drawing inferences about users and/or items without sufficient information.

	weight	height	size
count	162562.000000	191867.000000	192544.000000
mean	137.391709	65.310621	12.245175
std	21.899967	2.663480	8.494877
min	50.000000	54.000000	0.000000
25%	123.000000	63.000000	8.000000
50%	135.000000	65.000000	12.000000
75%	148.000000	67.000000	16.000000
max	300.000000	78.000000	58.000000

Figure 1: Basic summary statistics of numeric features. (Note: 'weight' is in pounds and 'height' is in inches)

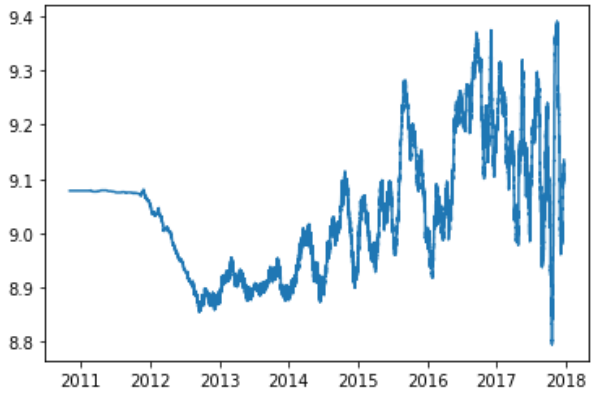

Figure 2: Average Rating sampled at 2 month intervals from the years 2011-2018

```
In [66]: (rent_runway_df.isnull().sum() / rent_runway_df.shape[0]) * 100
```

```
fit                0.000000
user_id            0.000000
bust size          9.561970
item_id            0.000000
weight            15.571506
rating             0.042588
rented for         0.005194
review_text        0.000000
body type          7.601899
review_summary     0.000000
category           0.000000
height             0.351608
size               0.000000
age                0.498587
review_date        0.000000
dtype: float64
```

Figure 3: Percentage of missing values within each column

```
sns.histplot(rent_runway_df['height'], bins = 25)
plt.gcf().set_size_inches(15, 8)
```

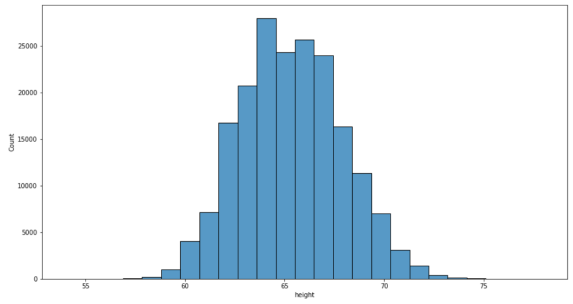

Figure 4: Histogram of the 'height' column

```
sns.histplot(x = rent_runway_df['weight'], bins = 30)
plt.gcf().set_size_inches(15, 8)
```

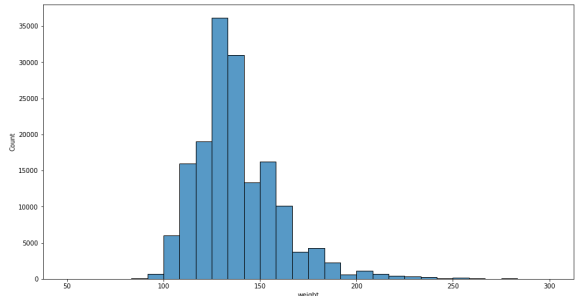

Figure 5: Histogram of the 'weight' column

Predictive Task

The predictive task we chose to do for this dataset was to build a recommender from latent factors. Our justification for this task was not only the significant amount of null data, but also a significant amount of single-item interactions. As a result, we did not want to simply use a similarity-based model (that does not use any features) due to their tendency to have cold-starts on this type of data.

For this task, we wanted to use features in the dataset as opposed to similarity due to the lack of interactions compared to users. And so the features used for this model are the latent factors of the dataset (or a sparse matrix of features).

To assess the validity of our model's predictions we evaluated our prediction using MSE (for ratings prediction) and AUC which is the measurement of valuing purchased items more than non-purchased items. The performance of the model is compared to baselines of linear regression and similarity.

Model

The model itself is a latent factor model created through the Tensorflow package. The model consists of a model class that contains the attributes of gamma (for both users and items) and beta (for items). The class contains functions for calculating the score as well as regularization that are used to rank the compatibility of a user to an item. The model is trained through gradient descent optimized on Adam, which is a learning algorithm that trains the model. Each iteration of learning updates the gamma weights and beta through the aforementioned functions.

Some of the models that we considered for comparison were simple models like Jaccard Similarity, and Linear Regression. An unsuccessful attempt was when we initially tried using linear regression based on various features such as date and text length. The MSE for rating prediction, however, was a sky-high 18.

A strength of the linear regression model is that it can be regularized to avoid overfitting. A weakness of the model is that linear regression performs poorly when looking at non-linear relationships. A strength of the Jaccard similarity is that it is good for ignoring duplicates when determining the similarity between sets. Some weaknesses of using the Jaccard similarity is that it does not consider term frequency and it lacks sensitivity to the differing sizes of sets. As similarity does not use any features at all, it also is prone to cold-starts.

As such we decided on a latent factor model, as it not only shares some strengths with the similarity model (being actually predictive) but also addresses the cold-start issue in similarity models.

There are in actuality two tasks: one for recommending items and the other for rating prediction. The rating prediction, though imperfect, was used to more easily compare the performance of the models. One of the potential issues we might have, however, is an issue with overfitting since the training set seems to perform better than the test set (a rating training MSE of 0.5 compared to 2 from the test set).

Literature

Our dataset was pulled off of Dr. Julian McAuley's Website[1]. Dr. McAuley himself got the data from a designer clothing renting website by the name of Rent the Runway. In the paper[1] "Decomposing Fit Semantics for Product Size Recommendation in Metric Spaces" by Misra, Wan, and McAuley, they utilized this dataset to predict the fit of items and recommend product size to users. They did this by implementing models that utilized metric learning and also used a latent factor model as well.

Another similar dataset dealing with clothing data from Amazon was studied by Charles Packer, Julian McAuley, and Arnau Ramisa[2]. In their paper, they used images from the clothing data to try and build a clothing recommender based on a user's individual "visual preference".

In their paper, “Hierarchical Fashion Graph Network for Personalized Outfit Recommendation” by Li, Wang, He, et al.[3], they worked on a similar task of outfit recommendation where they developed a new Hierarchical Fashion Graph Network (HFGN) model. The HFGN aimed to consider both user-outfit relationships and outfit-item relationships at the same time to create better recommendations to the user. The HFGN achieves this by utilizing information propagation from graph neural networks to get useful signals and uses these signals to create better outfits for the user. In the paper, they use a dataset pulled from another paper “POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion” by Chen, Huang, et al.[4], which is composed of click data from the online shopping platform Taobao. They found that their model performed better than various basic baselines such as Matrix Factorization and Visual Bayesian Personalized Ranking.

As mentioned before, personalized fashion recommendation is an increasingly popular task, and because of it, there have been quite a few methods developed to tackle such a task. Such methods include multilayer perceptron, recurrent neural network, k-nearest neighbor, and Bayesian networks [5]. Multilayer perceptron is an artificial neural network that has layers on neurons, and each neuron learns the weighted sum of its inputs and transmits signals to the next neuron. Recurrent neural networks are another deep learning algorithm that utilizes feed-forward neural networks. K-nearest neighbor is another classic classification algorithm and can be used for personalized fashion recommendation by looking at most similar items [5]. Bayesian networks are probabilistic graphical models that are made up of nodes and links that represent relationships between nodes. In the context of item recommendation, Bayesian networks can be used by looking at links between a “user's context” and their “personality” [5]. In addition to the algorithmic methods above, there are also similarity based methods including cosine, Jaccard, and Pearson similarity measures.

When looking at the results from the various papers, it is hard to compare our results as the baselines we compare our model to our different. However, in this paper “BPR: Bayesian Personalized Ranking from Implicit Feedback” by Rendle, et al. [6], they utilized Bayesian Personalized Ranking to two different baselines, matrix factorization and adaptive k-nearest neighbors, and found that by doing so, it improved performance in both cases. The adaptive k-nearest neighbor baseline also utilized a similarity function for the model, much like how we did and also found that their model performed better. This tells us that though similarity is useful within the recommendation space, it can be improved upon.

Results

```
recommendLFM(u, 5)
```

```
[(4.5223236, '126335'),
 (4.3555965, '123793'),
 (4.3396826, '174086'),
 (4.26921, '132738'),
 (4.265584, '145906')]
```

```
mostSimilar(u, 5)
```

```
[(0.009174311926605505, '227716'),
 (0.0018796992481203006, '168592'),
 (0.0, '999837'),
 (0.0, '999526'),
 (0.0, '998947')]
```

Figure 10: Outputs of the similarity and latent factor model

The parameters for our latent factor model included the length of sets by K=5, and the learning rate of 0.00001. The rating prediction model also had a mu value which was the average of the train set ratings. The parameters of some of the other models did not work as well. For example, the one-hot encoded date and text length linear regression model performed poorly. On the other hand, the feature representations for the latent factor model and similarity worked well.

We found that the latent-factor model is able to find good recommendations for most of the users, including for users with only one interaction. The similarity baseline, meanwhile, predictably had issues with users having only a single interaction. Take the above figure, for example, which is a recommendation for a user with only one interaction. The latent factor model gives several recommendations while the similarity model cannot give even more than two (after which items have a similarity score of zero).

Curiously, though, the similarity baseline performs better than the latent factor model in terms of raw performance data. The MSE of the ratings prediction for the similarity model was 2.13 compared to 2.18 with the latent factor model. Meanwhile, the AUC for the latent-factor model was about 0.83 compared to almost 1.0 for the similarity (0.999). However, this is misleading as it may be more due to the structure of the data and the way the similarity baseline predicts rating.

Rating predictions under the similarity baseline predict the mean of the data if there is not a good enough similarity. And since 68% of the data consists of single interaction data, the mean will be predicted most of the time. This is shown in the fact that the variance of the predicted values of the baseline is almost zero when compared to the variance of 0.3 of the Latent Factor model.

On the AUC point, the measure itself shows how well the model predicts unseen purchased data. On one hand, it seems that the similarity model performs very well, but the fact that it performs almost perfectly shows that this model is overfit to purchased data. And as seen in its performance in the examples given, it does not perform well on single interaction data, which is not good for a dataset with as many single interactions as the Rent the Runway dataset.

	fit	user_id	bust size	item_id	weight	rating	rented for	review_text	body type	review_summary	category	height	size	age
68	fit	245758	38d+	132738	NaN	8	party	This dress is gorgeous, but the front is VERY ...	NaN	I love RTR! I always feel amazing when I get a...	gown	5' 5"	16	NaN
243	small	423263	36d	132738	NaN	8	formal affair	I am 5'11 and I ordered a Long, however I had ...	athletic	I had a baby three weeks before NYE and this d...	gown	5' 10"	25	33
325	fit	857714	NaN	132738	200lbs	10	formal affair	the dress is so beautiful I im a plus size lad...	full bust	thank you RTR. everything was perfect!! my dres...	gown	5' 5"	36	31
628	fit	275613	NaN	132738	NaN	10	formal affair	This dress was gorgeous! I received tons of co...	full bust	Really fun, glamorous dress for charity event!	gown	5' 4"	20	38
719	small	842830	38c	132738	185lbs	8	formal affair	This dress was my original choice but ended up...	pear	Fun and glitzy	gown	5' 10"	29	36

	fit	user_id	bust size	item_id	weight	rating	rented for	review_text	body type	review_summary	category	height	size	age
4626	fit	986757	36b	999526	154lbs	8	work	Fits great, the structured fabric hides a mult...	pear	Beautiful dress but the itchy seams are TORTURE	dress	5' 10"	12	32
9752	large	435534	32c	999526	130lbs	10	party	I'd say it runs large -- size down. the materi...	pear	this dress is very unique and looks so much be...	dress	5' 3"	4	32
12157	large	585785	34a	999526	126lbs	8	everyday	This dress is a perfect a-line dress. The scub...	athletic	Cute, flattering... itchy	dress	5' 4"	4	40
12892	large	812733	NaN	999526	140lbs	10	party	It is very comfortable and only itchy occasion...	pear	I wore this to a rehearsal dinner and everyone...	dress	5' 7"	12	27

Figure 11: Top is from latent factor model, bottom is from similarity model. Note the difference between types of clothes. (The original user had just one interaction with a wedding dress).

Conclusion

For some datasets, needing to use data attributes may be necessary in some situations. In the case of the Rent the Runway dataset, the issue with cold-starts is visible and affects the quality of recommendations from a recommender that otherwise works well with high-interaction data.

Using a latent-factor model could help in these situations, even though they may come with some downsides of overfitting. Thus, it could be worthwhile to look into other models that may address these downsides such as perhaps a deep neural network.

References

[1] Rishabh Misra, Mengting Wan, and Julian McAuley. 2018. Decomposing fit semantics for product size recommendation in metric spaces. In *Proceedings of the 12th ACM Conference on Recommender Systems* (*RecSys '18*). Association for Computing Machinery, New York, NY, USA, 422–426. DOI:https://doi.org/10.1145/3240323.3240398

[2] Packer, Charles & McAuley, Julian & Ramisa, Arnau. 2018. Visually-Aware Personalized Recommendation using Interpretable Image Representations.

[3] Xingchen Li, Xiang Wang, Xiangnan He, Long Chen, Jun Xiao, and Tat-Seng Chua. 2020. Hierarchical Fashion Graph Network for Personalized Outfit Recommendation . In 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401080

[4] Wen Chen, Pipei Huang and Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, Binqiang Zhao. 2019. POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion. In Proceedings of Še 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Anchorage, AK, USA, August 4–8, 2019 (KDD '19), 9 pages. DOI: 10.1145/3292500.3330652

[5] Chakraborty, Samit, Md. S. Hoque, Naimur Rahman Jeem, Manik C. Biswas, Deepayan Bardhan, and Edgar Lobaton. 2021. "Fashion Recommendation Systems, Models and Methods: A Review" Informatics 8, no. 3: 49. https://doi.org/10.3390/informatics8030049

[6] Rendle, Steffen & Freudenthaler, Christoph & Gantner, Zeno & Schmidt-Thieme, Lars. (2012). BPR: Bayesian Personalized Ranking

from Implicit Feedback. Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, UAI 2009.