# DSCI 552 Project: Fashion Recommendations

University of Southern California

Kexin Zheng

## Abstract:

More and more recommendation systems appear in people's life, it can provide recommendation of products or other information according to the previous behavior of users. Lots of recommendation systems are designed based on Neural Network, however, it needs a large amount of training data and its model is sometimes hard to interpret. In this project, I tried to extract edge and color features from images of product to make recommendation. I used HOG to get the feature descriptor of images, and a combination of FastMap and K-means was applied to divide them into different clusters, EMD was finally used to calculate the color similarity. Final result was a csv file called "recommendArticles.csv" which stored customer ids and 12 recommended products for each of them.

## 1. Introduction

When we watch videos on YouTube, there are always several recommended videos under the video we are watching, and when we click on them again and again, the purpose of YouTube has been served. The same goes for clothing websites. When we are browsing the recommended products, we will always find something we want to buy.

Nowadays, a growing number of e-commerce sites are using recommendation systems to help consumers find products of interest[1]. The more the recommended product fits the consumer's preference, the more likely the customer is to buy it. A high-quality recommendation system can not only strength the relationship with their customers, but also offer higher returns. Nowadays, more and more researchers focus on the high-quality recommendation systems based on image using Neural Network. For example, Sun et al.[2] introduced a personalized clothing recommendation system using Siamese Convolutional Neural Network (SCNN) to measure the fashion style consistency of clothing items. While Batuhan et al.[3] provided a new clothing recommendation which does not need user's previous shopping act data using CNN.

However, Neural Network always requires a large number of data for training model, and their training models are hard to figure out.[4]

In this project, we made a personalized fashion recommendation for H&M group. To do so, we combined edge features and color features from images to make prediction. Histogram of Oriented Gradients (HOG), as a local feature extraction and description method, was utilized to obtain the edge descriptors. K-means algorithm was applied to cluster products based on their edge features because of its simplicity and high efficiency. Since K-means could provide a good performance on data that is numeric, continuous and low dimensional, but did not work well with higher dimension data, we needed to transform the high-dimensional data into low dimension to produce a better clustering result. FastMap, as a novel preprocess algorithm, provided an explicit Euclidean embedding in a near-linear time[5]. Thus, we used FastMap algorithm to embed all the images into 2D space, providing a good data form for K-means algorithm. Then, Earth Mover's Distance (EMD) was used to calculate color similarity. The final outcome was 12 recommended products for each customer.

## 2. Data and Methods

### 2.1 Dataset

In this project, the datasets from a Kaggle competition “H&M Personalized Fashion Recommendations” is used. It contains a folder with images of products, Figure 1 displays some sample images.


Figure 1. Display of some images of products

The csv file "articles.csv" stores product related information. There are 25 columns in this file, including lots of information about products, such as product name, type, color and department. Since the propose of project is to use as little extra information as possible, two attributes "article_id" and "product_type_name" are used in this project. In addition, the csv file "transactions_train.csv" provides the purchase history of customers. We used only the "customer_id" and "article_id" in this file. In this project, the purchase history of first 1000 data (first 302 customers) was utilized to make the recommendation. Figure 2 shows the content of these two files.

| | article_id | product_code | prod_name | product_type_no | product_type_name | product_group_name | graphical_appearance_no |
|---|---|---|---|---|---|---|---|
| 0 | 0108775015 | 108775 | Strap top | 253 | Vest top | Garment Upper body | 1010016 |
| 1 | 0108775044 | 108775 | Strap top | 253 | Vest top | Garment Upper body | 1010016 |
| 2 | 0108775051 | 108775 | Strap top (1) | 253 | Vest top | Garment Upper body | 1010017 |
| 3 | 0110065001 | 110065 | OP T-shirt (Idro) | 306 | Bra | Underwear | 1010016 |
| 4 | 0110065002 | 110065 | OP T-shirt (Idro) | 306 | Bra | Underwear | 1010016 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 105537 | 0953450001 | 953450 | 5pk regular Placement1 | 302 | Socks | Socks & Tights | 1010014 |
| 105538 | 0953763001 | 953763 | SPORT Malaga tank | 253 | Vest top | Garment Upper body | 1010016 |
| 105539 | 0956217002 | 956217 | Cartwheel dress | 265 | Dress | Garment Full body | 1010016 |
| 105540 | 0957375001 | 957375 | CLAIRE HAIR CLAW | 72 | Hair clip | Accessories | 1010016 |
| 105541 | 0959461001 | 959461 | Lounge dress | 265 | Dress | Garment Full body | 1010016 |

[105542 rows x 25 columns]

a

| | t_dat | customer_id | article_id | price | sales_channel_id |
|---|---|---|---|---|---|
| 0 | 2018-09-20 | 000058a12d5b43e67d225668fa1f8d618c13dc232df0cad8ffe7ad4a1091e318 | 0663713001 | 0.050831 | 2 |
| 1 | 2018-09-20 | 000058a12d5b43e67d225668fa1f8d618c13dc232df0cad8ffe7ad4a1091e318 | 0541518023 | 0.030492 | 2 |
| 2 | 2018-09-20 | 00007d2de826758b65a93dd24ce629ed66842531df6699338c5570910a014cc2 | 0505221004 | 0.015237 | 2 |
| 3 | 2018-09-20 | 00007d2de826758b65a93dd24ce629ed66842531df6699338c5570910a014cc2 | 0685687003 | 0.016932 | 2 |
| 4 | 2018-09-20 | 00007d2de826758b65a93dd24ce629ed66842531df6699338c5570910a014cc2 | 0685687004 | 0.016932 | 2 |
| ... | ... | ... | ... | ... | .. |
| 31788319 | 2020-09-22 | fff2282977442e327b45d8c89afde25617d00124d0f99982410630ac51314356 | 0929511001 | 0.059305 | 2 |
| 31788320 | 2020-09-22 | fff2282977442e327b45d8c89afde25617d00124d0f99982410630ac51314356 | 0891322004 | 0.042356 | 2 |
| 31788321 | 2020-09-22 | fff380805474b287b05cb2a7507b9a013482f7dd0bce0e6936f26ea7ecaa68a1 | 0918325001 | 0.043203 | 1 |
| 31788322 | 2020-09-22 | fff4d3a8b1f3b60af93e78c30a7cb4cf75edaf2590d3e593881ae6007d775f0f | 0833459002 | 0.006763 | 1 |
| 31788323 | 2020-09-22 | fffef3b6b73545df065b521e19f64bf6fe93bfd450ab20e02ce5d1e58a8f700b | 0898573003 | 0.033881 | 2 |

[31788324 rows x 5 columns]

b

Figure 2. The content in a) "articles.csv" b) "transactions_train.csv"

## 2.2 Methods

### 2.2.1 HOG

In 2005, the Histogram of Oriented Gradients (HOG) was firstly proposed by Dalal and Triggs. This feature descriptor can transform an image to a histogram of gradients which is widely used in computer vision and image process application for object detection.[6]

Firstly, all the images should be resized into the same shape to avoid some computation issues in the future steps. In this project, the shape of images are resized to $128 \times 64$. After resizing, the gradients for every pixel in the images are calculated. To be noted that gradients are the small changes in the x and y directions. Specifically, in the x direction, the gradient of one pixel is to subtract the value on the left from the value on the right, similarly, the gradient in the y direction is the subtraction of the value below the selected pixel from the value above the selected pixel. Figure 3 shows the filters for calculating gradient in x and y direction.

a

| | | |
|---|---|---|
| -1 | 0 | 1 |

b

| |
|---|
| -1 |
| 0 |
| 1 |

Figure 3. The filter for calculating gradient in a) the x direction b) the y direction

The magnitude and orientation are calculated using Pythagoras theorem. As shown in Figure 4, the magnitude of gradient $||\nabla f|| = \sqrt{G_x^2 + G_y^2}$, where $G_x$ and $G_y$ are the gradient in x and y direction. The orientation is given by $\emptyset = tan^{-1}(G_y/G_x)$.

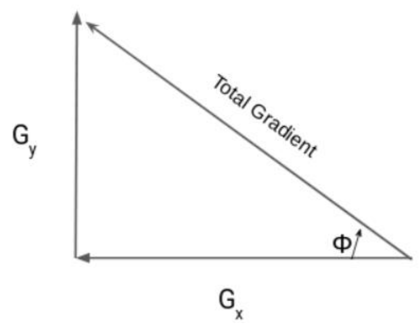


Figure 4. Schematic of calculation of the gradient and orientation using Pythagoras Theorem

A histogram is a plot that shows the frequency distribution of a set of continuous data. To get the histogram for the whole image, the image will be divided into small spatial regions which is known as "cells", the histogram of oriented gradients is computed for each cell. For each cell, the orientation of each pixel is checked and the corresponding magnitude is stored in the bins, in this project, the number of bins is 9, Figure 5 is a simple example of creating histogram in one cell. After creating the histogram for each cell, normalization is used for better invariance to illumination, shadowing, and edge contrast. For example, we divide the images into $8 \times 8$ cells and set the number of bins as 9, then combine four cells to create a $16 \times 16$ block, a matrix with shape $36 \times 1$ will be created as $V = [a_1, a_2, \dots, a_{36}]$, where $a_i$ means the $i^{th}$ value in the matrix. The normalized matrix is identified as $[a_1/K, a_2/K, \dots, a_{36}/K]$, where K is defined as $K = \sqrt{\sum_{i=1}^{36} a_i^2}$. In this case, 105 ($7 \times 15$) blocks are generated in an image with shape $64 \times 128$, the total number of features will be $105 \times 36 \times 1 = 3780$. [7]

Magnitude = 13.6
Orientation = 36

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Magnitude | | 13.6 | | | | | | | |
| Bin | 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |

Figure 5. A simple example of filling the matrix using magnitude and orientation

### 2.2.2 FastMap

FastMap is a data mining algorithm which is firstly introduce in 1995. It can embed complex objects such as DNA sequence and image into k-dimensional space. The algorithm is shown as follows:[4, 5]

Given a set of objects $O$, $D(i, j)$ represents the distance of objects $O_i$ and $O_j$. The farthest pair of objects $O_a$ and $O_a$ is firstly identified. To get the farthest pair, it starts with a random selection of object $O_a$. $O_b$, the farthest object away from $O_a$, is determined. It then reassigns $O_b$ as the object to start, and finds the farthest object again, until finding the farthest pair $O_a$ and $O_b$.

Once $O_a$ and $O_b$ are determined, each object $O_i$ can be considered as a part of triangle of $O_i$, $O_a$ and $O_b$ (Figure 6). The sides of triangle are defined as $d_{ai} = D(a, i)$, $d_{ib} = D(i, b)$, and $d_{ab} = D(a, b)$. According to the Law of cosines, the projection of $O_i$ is given by $x_i = (d_{ai}^2 + d_{ab}^2 - d_{ib}^2)/(2d_{ab})$.

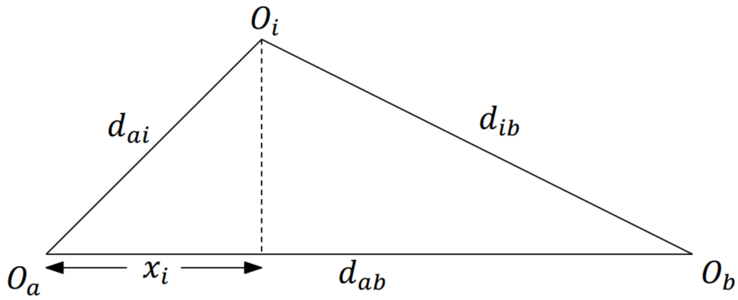


Figure 6. The law of cosines of the triangle.

FastMap sets the first coordinate of $p_i$, the embedding of $O_i$, equal to $x_i$. In this case, $x_a$ is set as 0 while $x_b$ is set as $d_{ab}$. The remaining K-1 iterations will repeat this process to obtain the other K-1 coordinates. However, the distance matrix used in each iteration need to be updated. In the second iteration, the above process should be completed on a hyperplane which is perpendicular to the line $\overline{O_a O_b}$, a conceptual construction of hyperplane is shown in Figure 7. In this case, the new distance between $O_i$ and $O_j$ is defined as $D_{new}(O_i', O_j')^2 = D(O_i, O_j)^2 - (x_i - x_j)^2$, where $O_i'$ and $O_j'$ is the projection of $O_i$ and $O_j$, $D_{new}(O_i', O_j')$ is the new distance function of $O_i'$ and $O_j'$. To be noted that $D_{new}(O_a', O_b') = 0$ since the hyperplane is perpendicular to line $\overline{O_a O_b}$.

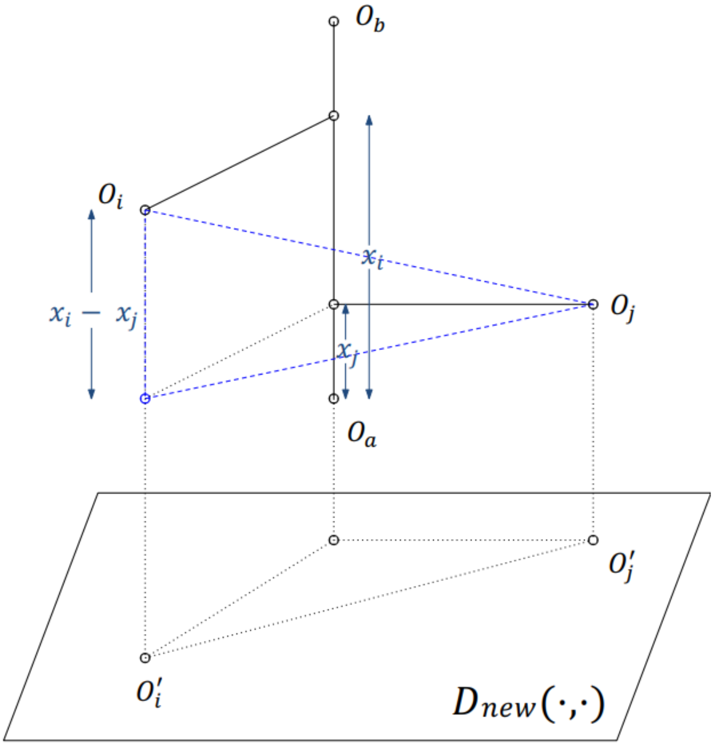


Figure 7. The conceptual construction of hyperplane for projecting object to a new space. This conceptual hyperplane is perpendicular to the line $\overline{O_a O_b}$ and is not explicitly.

### 2.2.3 K-means

K-means a typical clustering algorithm in data mining. It is unsupervised and has a wide range of applications in clustering the large set of data[8]. It provides a good performance of clustering in many regions. The following is how it works:

Given the data points and a pre-determined K value, where K represents the number of clusters. The centroids of K clusters are chose randomly, $\overline{\mu_i}$ represents the centroid of the $i^{th}$ cluster. The remaining K-means algorithm can be briefly divided into two steps: Assignment step and summarization step. In the assignment step, each data point is assigned to its nearest centroid. Euclidean distance is used to calculate the distance between data point $x_i$ and centroid $\mu_j$, cluster $c_i = \underset{j}{argmax}||x_i - \mu_j||^2$, where $c_i$ represents the cluster which $x_i$ belongs to. Then the next step is to recalculate the centroid of each cluster based on the data points belonging to this cluster. This iterative process continues repeatedly until the assignment no longer changes.

### 2.2.4 EMD

The Earth Mover's Distance (EMD), also known as Wasserstein metric, is a method to evaluate dissimilarity between two multi-dimensional distributions. It is firstly introduced by Yossi et al. in 2000[9]. It is widely used in content-based image retrieval to calculate the distance of color histograms between two images.

The distribution of color space of image is firstly converted to the CIE-Lab color space. In the ground distance $d_{ij} = 1 - e^{-\alpha||p_i - q_j||}$, $|| * ||$ is $L_2$-norm, $\alpha = ||[\sigma_1, \sigma_2, \dots, \sigma_{dim}]^T||$, where $\sigma_i$ is the standard deviation of the $i^{th}$ dimension components of the features from the overall distribution of all images in the database.

Given $P = \{(p_{1,}w_{p_1}), \dots, (p_{m,}w_{p_m})\}$ as the first signature with m clusters, $p_i$ represents the cluster and $w_{p_i}$ is the weight of this cluster. Similarly, $Q = \{(q_{1,}w_{q_1}), \dots, (q_{n,}w_{q_n})\}$ as the second signature with n clusters. Ground distance matrix $D = [d_{ij}]$, where $d_{ij}$ is the ground distance between cluster $p_i$ and $q_j$.

What we want to find is a flow $F = [f_{ij}]$, where $f_{ij}$ is the flow between $p_i$ and $q_j$, such that the overall cost $WORK(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij}$ reaches the minimum. This minimization have the following constraints:

$$f_{ij} \geq 0 \quad 1 \leq i \leq m, 1 \leq j \leq n$$

$$\sum_{j=1}^{n} f_{ij} \leq w_p \quad 1 \leq j \leq n$$

$$\sum_{i=1}^{m} f_{ij} \leq w_q \quad 1 \leq i \leq m$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = \min\left(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{q_j}\right)$$

After finding the flow F, the earth mover's distance is defined as

$$EMD(P, Q) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij} \,/\, \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}.$$

In this project, the python library pyemd is used to calculate the Earth Mover's Distance.

## 3. Results

In this project, the purchase history of 302 customers was used to make a fashion recommendation. In order to make it easier and more efficient to use these data in the future, the customer ids and a list of article ids they purchased were read and stored in a new csv file called "purchase.csv" (Figure 8).

To get the similar products of a given product by image, multiple algorithms were used. In general, we firstly extracted the edge descriptor of each image and embedding these images into a 2D space using FastMap algorithm. After getting the 2D coordinate for every image, these images were divided into different clusters by K-means algorithm. Finally, EMD was utilized to find the similarity of color, and finally provided 12 recommended products for every customer.

```
                                         customer_id                                        article_id
0    000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...                 ['0663713001', '0541518023']
1    00007d2de826758b65a93dd24ce629ed66842531df6699...  ['0505221004', '0685687003', '0685687004', '06...
2    00083cda041544b2fbb0e0d2905ad17da7cf1007526fb4...  ['0688873012', '0501323011', '0598859003', '06...
3    0008968c0d451dbc5a9968da03196fe20051965edde741...                 ['0531310002', '0529841001']
4    000aa7f0dc06cd7174389e76c9e132a67860c5f65f9706...  ['0501820043', '0501820043', '0674681001', '06...
..                                                 ...                                               ...
297  057e275b214b3d97a8caa75a3b802bf2ffda4d490189a1...                               ['0504154016']
298  05887aca6a1742c97497dded62e474cc10b5c5d783eaff...                               ['0691012001']
299  0592e247fc4388fe73eaacb8d8577e5d817b4b0c7e679c...  ['0636421003', '0583558001', '0661308001', '05...
300  05943a58bd172641b80919a9bdf14012df940800bc74d0...  ['0648940001', '0661794001', '0661794001', '06...
301  059d4230aeb70d519fb5e5e5ec8f5efc6c3eaad65b06af...                               ['0516903005']

[302 rows x 2 columns]
```

Figure 8. A sample display of "purchase.csv"

### Get feature descriptors

Before classification of images based on their edge descriptors, we firstly separated them into 131 groups according to the given product type. It was because that there were about 106k products which was too big for the memory. For each product type, images were resized into $128 \times 64$. The feature descriptors of images were obtained using HOG method. Python function hog() from skimage.feature was used to implement HOG method, the attribute "orientations" was set to 9 which meant the number of bins in the histogram, the size of the cell ("pixels_per_cell") was assigned to $4 \times 4$, while attribute "cells_per_block" was set to $1 \times 1$. In this case, the feature descriptor had 4608 ($32 \times 16 \times 9$) features. Figure 9 shown a sample image after HOG.


Figure 9. Process of converting an image into HOG image

**Divide into different clusters**

To transforming the high dimension of data into low dimension, FastMap was performed. The distance $D(O_i, O_j)$, which represented the distance between image i and image j, was calculated through Euclidean distance after flattening the HOG images matrix. The value of K was taken as 2 which meant that images were embedded into 2D space. After FastMap algorithm, an array with shape $N \times 2$ was created for each product type, where N is the number of images in this type. K-means algorithm was then used to separated them into different clusters. Shown as Figure 10, the article ids and cluster number was stored in csv files.

```
      article_id  cluster
0     0116379047       12
1     0145872051        9
2     0163734002        7
3     0163734054        7
4     0234622003        5
...          ...      ...
4147  0938622001        2
4148  0941310001       11
4149  0944989001        9
4150  0947599001        0
4151  0952938001        5

[4152 rows x 2 columns]
```

Figure 10. The contents of csv file “top.csv”, “top.csv” stores the id and cluster number of products whose type is top.

## Get images with similar color

To find the similar articles according to the color, we read all the article ids with the same type and cluster as the purchased item, resized them to shape $32 \times 32$ to reduce computation, and convert them into CIE Lab so that short Euclidean distances correlated strongly with human color discrimination performance, despite being paired colors on a neutral background. In this project, python library pyemd was used to calculate the EMD between two images. Figure 11 gives a simple flow chart of this process.

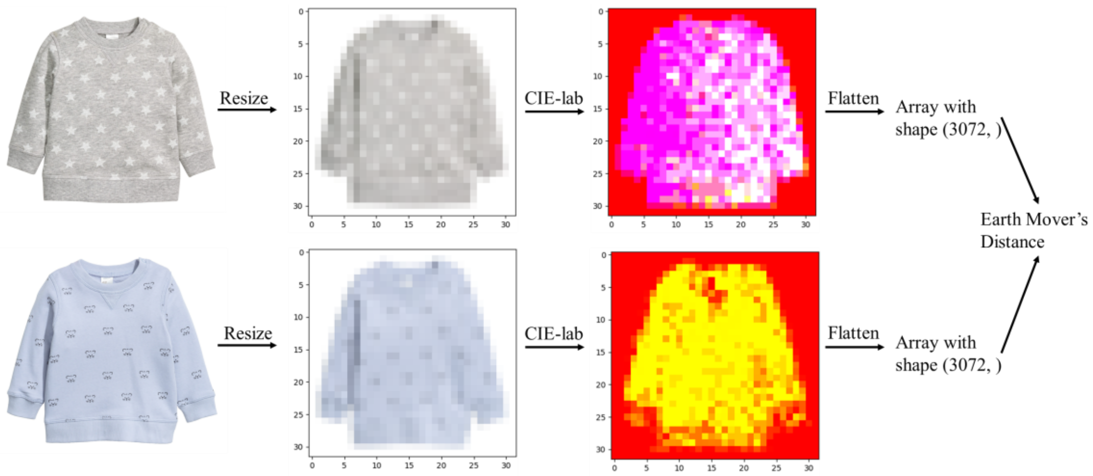


Figure 11. Flow chart of preprocess of data when calculating EMD

## Logic of recommendation

When making recommendation, the general idea was to read every article image and picked the most similar ones, if the article customer bought had no corresponding image, we chose to ignore it. There would be several situations to be considered:

When one customer only bought one or two products and the total number of similar products were less than 12, we would recommend him or her the most popular products to supplement it. The most popular products were determined by counting which products these customers bought the most;

When a customer bought more than 12 products, we would find 1 similar article for each article, and select the first 12 products to recommend. If one product were bought more than once, or there were some items similar to each other, the recommendation logic would ignore the duplicated items to make sure that 12 products were different from each other.

The final result was saved in a csv file called “recommendArticles.csv” (Figure 12). It had all customer ids and the recommended article ids for each of them. Figure 13 is one of the results.

| | customer | predicition |
|---|---|---|
| 0 | 000058a12d5b43e67d225668fa1f8d618c13dc232df0ca... | 0783830001 0566428001 0650037001 0781842002 07... |
| 1 | 00007d2de826758b65a93dd24ce629ed66842531df6699... | 0537346004 0568862004 0698048002 0624066008 07... |
| 2 | 00083cda041544b2fbb0e0d2905ad17da7cf1007526fb4... | 0764228001 0627339002 0707225001 0594264003 07... |
| 3 | 0008968c0d451dbc5a9968da03196fe20051965edde741... | 0504151003 0504152002 0539387006 0519856002 06... |
| 4 | 000aa7f0dc06cd7174389e76c9e132a67860c5f65f9706... | 0732423001 0704119001 0913688001 0695170004 09... |
| .. | ... | ... |
| 297 | 057e275b214b3d97a8caa75a3b802bf2ffda4d490189a1... | 0539366003 0554141006 0568869003 0547300002 05... |
| 298 | 05887aca6a1742c97497dded62e474cc10b5c5d783eaff... | 0804798001 0710876020 0744306019 0654863002 07... |
| 299 | 0592e247fc4388fe73eaacb8d8577e5d817b4b0c7e679c... | 0846347003 0664340001 0771274001 0731523002 08... |
| 300 | 05943a58bd172641b80919a9bdf14012df940800bc74d0... | 0650653003 0630299001 0691396003 0673711002 08... |
| 301 | 059d4230aeb70d519fb5e5e5ec8f5efc6c3eaad65b06af... | 0663588001 0657285002 0626890004 0596517001 06... |

[302 rows x 2 columns]

Figure 12. Contents of “recommend.csv”

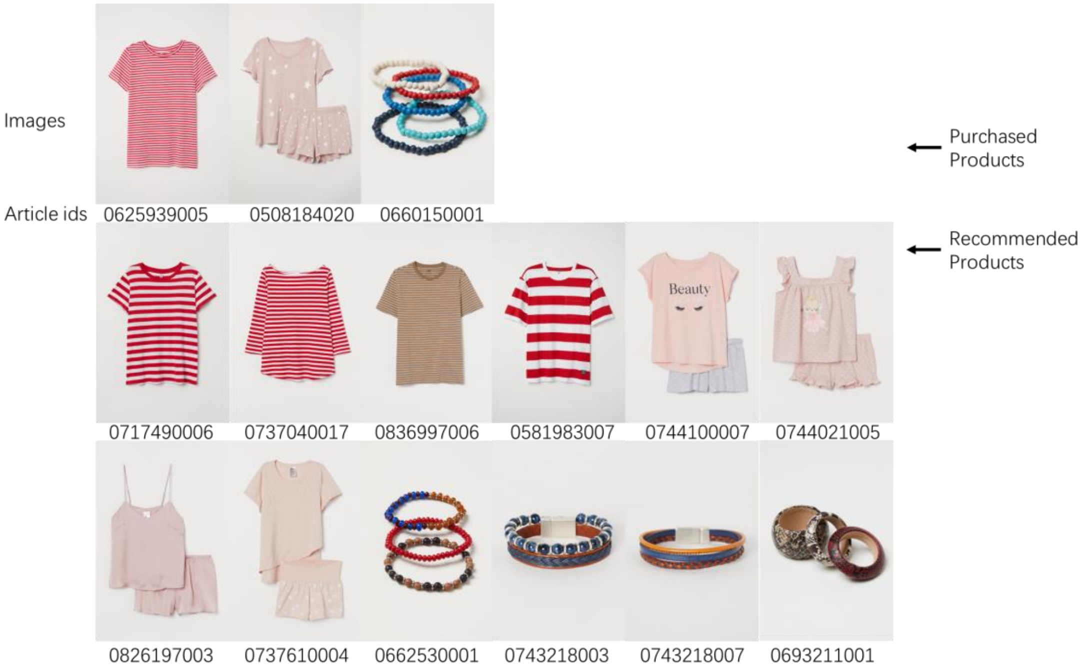


Figure 13. Images of products bought by a customer and recommendations. (customer id = 005c9fb2ba6c49b2098a662f64a9124ef95cbec5fcf4ebdb4dcbaaf83f979c51)

## 4. Conclusion

In this project, a fashion recommendation program was built. Firstly, it combined FastMap and K-means to divided products into similar groups according to their HOG images, then used EMD to obtain the products with similar colors, and finally picked

12 products for each customer. We could found that the final result gave a good recommendation which means that the combination of algorithms in this project shown a good performance. However, there are also some places which need to be improved. Firstly, when using FastMap to embed 10k data into 2D algorithm, although it ran fast, it still required several hours due to the volume of data. So in the next step, I would like to investigate how to speed up this process. Besides, the logic of the recommendation should be improved. For example, if the number of dresses was larger than that of trousers, we need to recommend more dresses than trousers. Thus, my idea is to set weight so that we could focus more on the category with the most purchases.

The codes of this project are post in the GitHub, the website is https://github.com/kexinzheng05/DSCI552-project.

# References


[1] Schafer, J.B., Konstan, J.A. and Riedl, J. E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery* 5**,** 115–153 (2001). https://doi.org/10.1023/A:1009804230409

[2] Sun, GL., Cheng, ZQ., Wu, X. et al. Personalized clothing recommendation combining user social circle and fashion style consistency. *Multimed Tools Appl* 77, 17731–17754 (2018). https://doi.org/10.1007/s11042-017-5245-1

[3] B. AŞIROĞLU, M. İ. ATALAY, A. BALKAYA, E. TÜZÜNKAN, M. Dağtekin and T. ENSARİ, "Smart Clothing Recommendation System with Deep Learning," *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2019, pp. 1-4, doi: 10.1109/ISMSIT.2019.8932738.

[4] M. C. A. White, K. Sharma, A. Li, T. K. S. Kumar, and N. Nakata, FastMapSVM: Classifying Complex Objects Using the FastMap Algorithm and Support-Vector Machines. *arXiv,* 2022, doi: https://doi.org/10.48550/arXiv.2204.05112

[5] L. Cohen, T.l Uras, S. Jahangiri, A. Arunasalam, S. Koenig, and T. K. S. Kumar, The FastMap Algorithm for Shortest Path Computations, *arXiv,* 2017, doi: https://doi.org/10.48550/arXiv.1706.02792

[6] A. Waheed, How to Apply HOG Feature Extraction in Python, *PythonCode*, Dec 2021, https://www.thepythoncode.com/article/hog-feature-extraction-in-python.

[7] A. Singh, Feature Engineering for Images: A Valuable Introduction to the HOG Feature Descriptor. *Analytics Vidhya*, 4 Sep, 2019, https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor.

[8] S. Na, L. Xumin and G. Yong, "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm," *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, 2010, pp. 63-67, doi: 10.1109/IITSI.2010.74.

[9] Rubner, Y., Tomasi, C. and Guibas, L.J. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* 40, 99–121 (2000). https://doi.org/10.1023/A:1026543900054