NEW YORK UNIVERSITY

GROUP PROJECT

Apparel Classification

Author:

Yuting Gui

Xinyi Gong

Minqing Zhuang

Supervisor:

Kurt Miller

A project submitted in fulfillment of the requirements

for the course Machine Learning

Center of Data Science

May 13, 2016

Contents

Introduction

Clothing serves for much more than just protection and covering. It is a means of communication to reflect social status, lifestyles, or even fashion sense. Therefore, a huge amount and various styles of clothes have been produced to satisfy people's different taste.

Comparing to traditional in-store shopping, nowadays, people are more likely to buy clothes online. The main reason is that online shops provide tons of resources. In addition, there are numbers of e-commerce websites that provide elegant shopping environment embedding with powerful filters and search engines for target clothes, which make the shopping process very convenient and efficient. Shopping websites receive large amount of clothes pictures everyday. One of their concerns is that how to efficiently classify clothes pictures with minor human work.

The objective of our project is to classify clothes by purely scanning and analyzing clothes pictures. Furthermore, we will provide better filters to improve customer experience of the online shopping websites.

Data Preparation

2.1 Data Collection

Polyvore.com is a community-powered social commerce website [1]. It provides numerous standard and uniformed clothes images for community members to create image collages and share with others. The collection of clothes image data from polyvore.com exploited the Python package *Selenium* and *BeautifulSoup*. Selenium served as the automates browser and BeautifulSoup is used as HTML parser. The url links we parsed were given by searching particular keywords, which we used to label the images, on polyvore.com. During the process of scraping images, the captions and labels has been scrapped to our local machines as well.

The reason we chose Polyvore is that pictures from it are standard and uniformed, such that they all have white background, consistent lighting conditions, and similar image scales. It is a good start point of following image preprocessing. The other fact of polyvore images is that they has two kinds of clothes pictures in general, with and without human model.

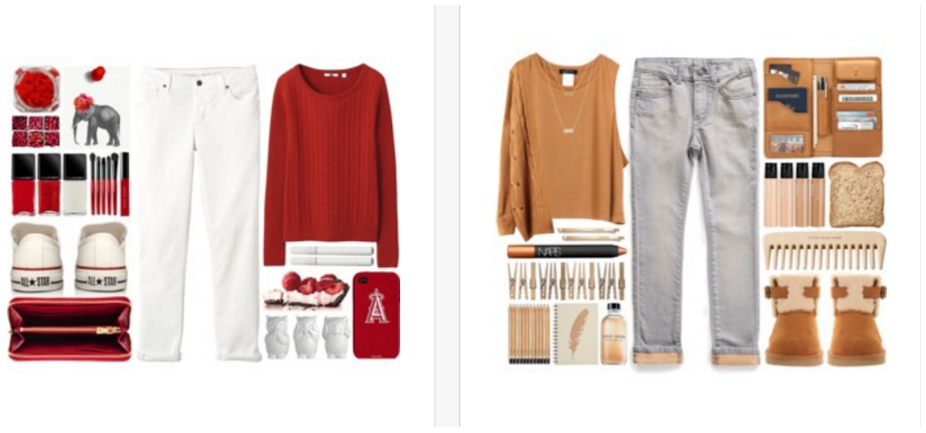

FIGURE 2.1: picture collages from polyvore

2.2 Label and Split

We first labelled data into four categories: 'short sleeve', 'long sleeve', 'skirt', 'pant'. Each category has approximately 500 pictures. The models and results are shown in Experiment part. The accuracy with this labelled data is as low as randomly choosen (about 25%). In order to find reasons, we build models based on same data labelled 'with human model', and 'without human model'. The accuracy was higher than 90%, which suggested highly difference between these two categories. Therefore, the data finally were labelled as five categories: 'short sleeve without model', 'long sleeve without model', 'skirt without model', 'pant without model', and 'clothes with model'.

The other issue regarding to the dataset was the "wrong" pictures within each category. For example, pant pictures appeared in skirt category. In order to solve this problem, the categories were refined by taking advantages of its caption. For example, we removed the picture from skirt category if the caption of the picture has keywords like 'pand' or 't-shirts'. Finally, we had 1920 pictures in total. The data then was randomly split to train, validation, and test sets, with 1227(64%), 306(16%), 387(20%) pictures respectively. The reason that we collected small amount of images is that we don't want to spend long time to train one model. In this project we will focus on distinguishing which kind of model is more suitable in clothes image data.

2.3 Data preprocessing

The images have been resized from 3*300*300 into 3*128*128 resolution (RGB color, pixels, pixels). Based on resized images, we generated the rotate images: each image in dataset has been randomly rotate from -30 to 30 degree. The reason for generating rotate images is that we want the classifiers learn more invariant features, besides only memorizing the position of colored pixels. We also generated the grayscale images from RGB data, in order to feed in specific models, which would produce images in 1*128*128 scale. The function we used to transfer RGB to grayscale is $Y = 0.2126 * R + 0.7152 * G + 0.0722 * B$. The original data, grayscale data, and rotate data are shows in Figure 2.2.

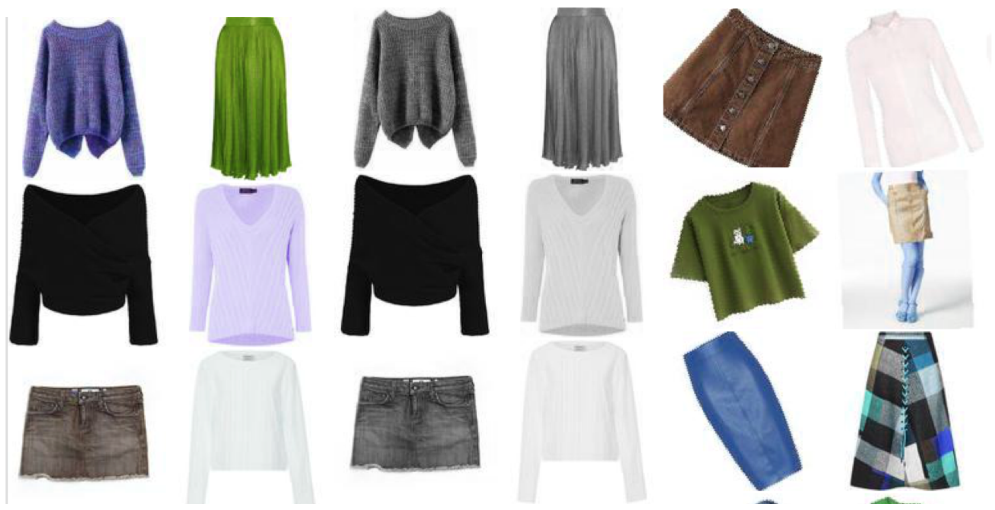
FIGURE 2.2: Sample of original, grayscale, and rotate data

Model

3.1 local Binary Patterns

Local Binary Patterns [2] is an important feature descriptor that is used in computer vision for texture matching. LBP Descriptor aquires graycsale images. For each pixel in the grayscale images, a neighborhood is selected around the current pixel. A binary number of each neighborhood is calculated with respect to the central pixel: if neighbor's value is greater than central pixel, then this neighbor assigned 1, otherwise, assigned 0. Therefore each central pixel is associated with a list of binary number. Then the list of binary number is converted to a decimal number stored in the corresponding location in the LBP mask. The converting process demonstrated in Figure 3.1. Then transfered the LBP mask of image to a histogram.

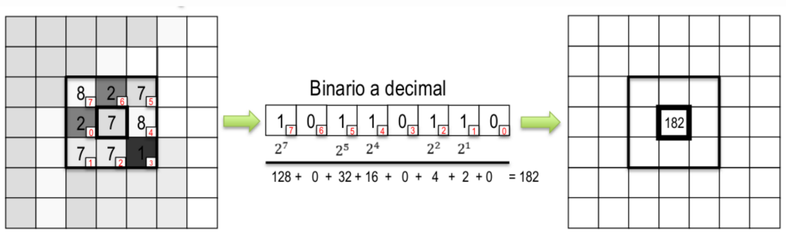

FIGURE 3.1: LBP Mask Calculation [7]

Baseline model: Compute the histogram to all of the trainset and testset images. After that, compare each testset histogram with trainset histograms using chi-square distance. For each testset image, we take the label of the image in trainset that has smallest chi-square distance as predicted label. The prediction accuracy shows in Top1 column in table 3.1. We also store the 5 smallest distance images and the 20 smallest distance images in trainset, and take the mode of their labels as the predicted label. The results shows in Top5, and Top20 columns in table 3.1. The best accuracy **69%** gives by taking 15 neighbors in calculating LBP number, and using the mode label of 5 smallest distance image in trainset as predicted label.

Neighbors Number	**Top1**	**Top5**	**Top20**
8	65.63%	67.44%	63.82%
15	67.18%	**69.00%**	68.99%
24	66.15%	66.92%	67.18%

TABLE 3.1: LBP accuracy

LBP with linear classifiers:

We also trained linear classifier with LBP texture features. We used the histogram of LBP mask as features. The number of neighbors used to compute LBP are 8, 15, and 24. Each image is now represented by an array of its normalized histogram. Each image has 26 features with 8 neighbors, 24 features with 15 neighbors, and 42 features with 24 neighbors. We try Random Forest, Decision Tree Classifier, Logistic Regression, and One versus Rest with Linear SVM, and employ 5-fold cross validation to tune hyperparameters. The best performance given by 8 neighbors, Training, test, and validation errors for each model and model's optimal hyperparameters show below.

8 Neighbors	**Train acc**	**Valid acc**	**Test acc**
LR(l1,λ = 100)	68.5%	68.4%	-
One v Rest Linear SVM (l2, λ=100)	65.2%	66.00%	-
DT(MaxDepth = 6)	74.5%	61.6%	-
RF(trees = 16)	99.9%	**70.7%**	**66.9%**

TABLE 3.2: LBP accuracy

3.2 Convolutional Neural Network

Convolutional neural network (CNN) [3] is a non-linear classifier. It has similar structure with ordinary neural network: receive an image input and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons. Unlike ordinary neural network, CNN have neurons arranged in 3 dimensions. In our dataset, the input image has dimension 128×128×3, and the final output should have dimension 1×1×5. Each number in the output vector represents the score that this image belongs to the category.

The specific models structure shows below. We build CNN on both original and rotate dataset. In the following model structure part, CONV represents convolutional layer, POOL represents maxpooling layer, RELU represents rectifier linear unit layer, BN represents batch normalization layer, DO represents Dropout layer, FC represents fully connected layer.

The learning objective is minimizing cross entropy loss using stocastic gradient descent method. And use accuracy to measure the performance. The model runs 100 epoch to learn the weights of each neurons. The final results shows in table 3.3 and table 3.4.

CNN models	**Train acc**	**Valid acc**	**Test acc**
Model1:[CONV-RELU-POOL-DO]*3-RELU-FC	68.01%	34.67%	-
Model2:[CONV-RELU-POOL]*4-DO-BN-DO-FC	96.13%	91.33%	-
Model3:[CONV-RELU-POOL]*4-BN-FC	98.35%	**92.33%**	**90.40%**

TABLE 3.3: Last epoch CNN accuracy on original data

CNN models	**Train acc**	**Valid acc**	**Test acc**
Model4:[CONV-RELU-POOL]*4-BN-RELU-FC	92.02%	**84%**	**86.66%**
Model5:[CONV-RELU-POOL]*2-BN-RELU-FC	88.65%	83.33%	-

TABLE 3.4: Last epoch CNN accuracy on rotate data

The reason that Model1 is worse than Model2,3 is that the final CONV layer in Model1 has 128 filters, whereas in Model2 and Model3, the final CONV layer has 256 filters. The slightly difference results between Model2 and Model3 is that Model has dropout layer to prevent overfitting, but Model3 is not. The other interesting fact we found is that the batch normalization layer is extremely useful in analyzing our dataset. If we drop the BN layer, the accuracy in all train, valid and test set were not exceed 30%.

3.3 Linear Classifiers

3.3.1 Data Manipulation

From the view of machine learning, linear classifiers take the value of linear combination of characteristics as classification decision. In order to find linear features of our dataset, we exploited several linear classifiers to naively learn clothing images. By the word "naively", we mean that every image has been reshaped from 3*128*128 to 49152. For following linear classifier, every input image in dataset goes to a sample with 49152 features, and the dataset becomes two-dimensional (length of dataset $\times$ 49152).

The dataset has been split into train, validation, test, with 1227, 306, 387 pictures respectively. In this part of training linear classifiers, we combain train and validation set together, and leave test alone. By doing that, we are able to apply cross-validation method on our combained larger train set. As we have a large more features than samples, the models expose to the problem of overfitting. Therefore, regularization has been applied to some of the linear classifiers to prevent the overfitting. Regularization is a technique to penalize complex models, main regularizers include L1 and L2 norm. We expect better performance with applying regularization in models.

3.3.2 Linear models

The models we build include: Logistic Regression, Multiclass linear SVM, Multinomial Naïve Bayes, Decision Tree, Random Forest, and AdaBoost with Decision Tree based on original data and rotate data. In order to get a general idea of the performance of each model, we applied cross validation for the combained original data with default parameters given my package sklearn. We chose 5 folds for the cross validation, which is separating all the training dataset to be in 5 blocks, and for the ith round, use the i^{th} block as the validation set. The accuracy of 5 models discussed above is shown in figure 3.2. the darker the color, the more accuracy the model achieves.

From figure 3.2, logistic regression, random forests and one versus rest with linear SVM have higher accuracy with default parameters by sklearn. So we tune the parameters of

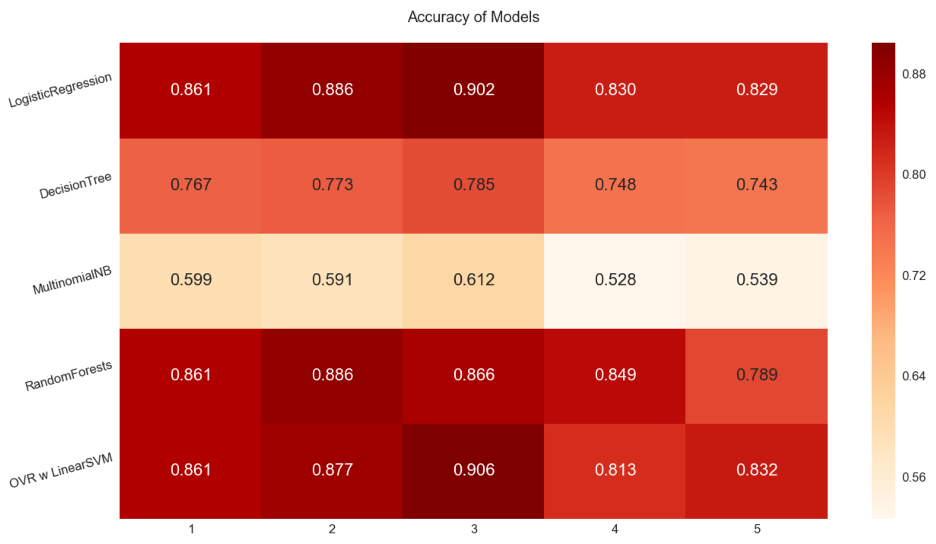

FIGURE 3.2: Linear classifier accuracy on validation set in 5 folds cross-validation

those models on original data and rotate data. Final models accuracy for test set show in table 3.5 and 3.6.

In table 3.5 and 3.6, LR represents logistic regression, RF represents random forest, One-Rest SVM represents one vs rest SVM We use validation accuracy to choose model, for the model with best performance in validation set, calculate the testset accuracy.

Linear models	**Train acc**	**Valid acc**	**Test acc**
LR(l2, $\lambda = 0.5$)	99.92%	82.68%	-
RF(trees = 35)	99.91%	**87.25%**	**87.60%**
One - Rest SVM(l2, $\lambda = 0.5$, square hinge loss)	99.72%	81.38%	-

TABLE 3.5: Linear classifier accuracy on original dataset

Linear models	**Train acc**	**Valid acc**	**Test acc**
LR(l1, $\lambda = 0.1$)	100%	**72.2%**	**75.19%**
RF(trees = 35)	99.34%	62.9%	-
One - Rest SVM(l2, $\lambda = 0.001$)	100%	70.26%	-

TABLE 3.6: Linear classifier accuracy on rotate dataset

Analysis

4.1 Model Analysis

The accuracy for both linear and non-linear models on original dataset are high, but accuracy for rotate data are lower. The non-linear model decreased in smaller amount in accuracy than the linear model. The reasons are as follow: our linear classifier took groups of vectors as input, and they could be regarded as colored pixels position information. Therefore, linear features in some sense are the position information. When we ramdomly rotated the images, the position of colored pixels for each image changes, so that the linear classifier performance went much worse. However, the non-linear classifier CNN was learning more invariant feature such as edge. In Figure 4.1, CNN model was taking look of a pant. It transform a pant picture to following activation map. Then it use CNN features to filter pictures in activation map, and see whether features match pictures.

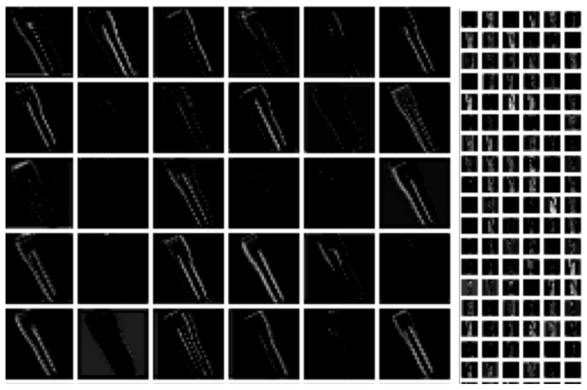
FIGURE 4.1: CNN filters in the 3rd and 4th CONV layer

Linear classifiers which has roughly bad performance:

Naive Bayes: we found that naive bayes are not suitable for the image data, it is more suitable the dataset has discrete feature such as natural language processing problem, however, our image features are continuous in 0-255 rgb space.

SVM with RBF kernel: SVM with linear kernel works perfect on our dataset, but SVM with RBF kernel got performance as bad as 25% accuracy. The first reason is that our data are linearly separable. Most of the linear classifier works well have shown this point. The second reason is that we have 49152 features whereas only about 1500 samples, so that mapping the data to higher dimension does not really help improve the performance.

Non-linear classifiers which has roughly bad performance:

Convolution layer not has 256 filters: For our dataset, the final convolution layer should has at least 256 filters, otherwise, it is not deep enough to get useful filters. However, the number of convolution layer doesn't matter. We tried two, three, four convolution-pooling layers, keep every other layers same. At least the final convolution layer has 256 filters, the accuracy on validation set will exceed 90% on original dataset.

Not has Batch Normalization(BN) layer: The BN technique first was proposed to accelerate deep network [8]. In our dataset, BN layer is essential to get good performance. We compare the train, valid accuracy on two models with only difference in BN layers, and run them with 100 epoch on rotate dataset. The results shows in table 4.1:

CNN models	Train acc	Valid acc
Model1:[CONV-RELU-POOL]*2-DO-RELU-FC	24.59%	24.67%
Model2:[CONV-RELU-POOL]*2-DO-BN-RELU-FC	88.65%	83.33%

TABLE 4.1: Last epoch CNN accuracy

4.2 Error Analysis

Why high accuracy: The dataset has some pictures that are really similar each other. For example, gray t-shirts, they looks very similar. It is possible that similar pictures being split to train, valid, test sets, which makes the classification easier. In addition, the background of images are white, so that it is easy to get position information of colored pixels.

Which pictures make classifier confused: In order to see what kind of picture got misclassified, we extracted the misclassified pictures from the dataset. We find that all misclassified images are clothing with long sleeve, and some pictures are misclassified by multiple models. The reason is that they really look like pictures in other classes. Following are two examples of pictures that are misclassified by more than one models and very similar to clothing in other classes. The picture on the left in Figure 4.2 should be classified as long sleeve but misclassified as pants, and the picture on the right misclassified as skirt.

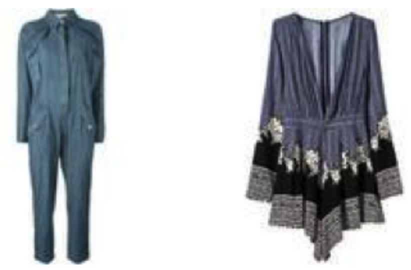
FIGURE 4.2: misclassified pictures

Conclusion

The clothes image has linear features if they are in right direction. The linear features are easy to learn. On the other hand, clothes image also has non-linear features such as edge, pattern features, which is easy to learn by CNN.

Based on all models' performance, the convolutional network gives best performance on test set, and it doesn't tend to overfit dataset by using dropout regularization method. It is surprising that the linear classifiers have high accuracies in test set. It may because we have reshaped all the input images to the same shape and all the images are clean and well-organized. Specifically, images have values on the same positions so that it is easy for linear classifiers to predict. In addition, based on the parameters we chose, within the framework of multi-classification, linear classifiers perform better with one versus rest method. LBP, served as a pattern descriptor, did well on the clothing dataset.

Although CNN outperforms linear classifier in our dataset, it takes much longer time to train the model. When we get large dataset, it will take a lot of time to train a good CNN model.

In the following development of the models, we can use other feature extraction method such as SURF [4], SSD [5], as well as color information in L*a*b space [6]. Then linear classifier can be build on feature extracted by above methods.

Bibliography

[1] https://en.wikipedia.org/wiki/Polyvore

[2] T. Ojala, M. Pietikäinen, and D. Harwood (1996), "A Comparative Study of Texture Measures with Classification Based on Feature Distributions", Pattern Recognition, vol. 29, pp. 51-59

[3] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner(1998), "Gradient-Based Learning Applied to Document Recognition", Proceedings of the IEEE, 86(11):2278-2324

[4] H. Bay, T. Tuytelaars, L. Van Gool: SURF: Speeded Up Robust Features. ICCV (2006)

[5] E. Shechtman, M. Irani,: Matching Local Self-Similarities across Images and Videos. CVPR (2007)

[6] L. Bossard, M. Dantone, C. Leistner, C. Wengert, T. Quack, L. Van Gool: Apparel Classification with Style

[7] http://hanzratech.in/2015/05/30/local-binary-patterns.html

[8] S. Ioffe, C. Szegedy:Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift